Available online at www.sciencedirect.com
ScienceDirect
Comput. Methods Appl. Mech. Engrg. 346 (2019) 216–241
Computer methods in applied mechanics and engineering
www.elsevier.com/locate/cma

Meta-modeling game for deriving theory-consistent, microstructure-based traction–separation laws via deep reinforcement learning

Kun Wang, WaiChing Sun*

Department of Civil Engineering and Engineering Mechanics, Columbia University, 614 SW Mudd, Mail Code: 4709, New York, NY 10027, United States

Received 4 September 2018; received in revised form 25 November 2018; accepted 26 November 2018
Available online 7 December 2018

Abstract

This paper presents a new meta-modeling framework that employs deep reinforcement learning (DRL) to generate mechanical constitutive models for interfaces. The constitutive models are conceptualized as information flow in directed graphs. The process of writing constitutive models is simplified as a sequence of forming graph edges with the goal of maximizing the model score (a function of accuracy, robustness and forward prediction quality). Thus meta-modeling can be formulated as a Markov decision process with well-defined states, actions, rules, objective functions and rewards. By using neural networks to estimate policies and state values, the computer agent is able to efficiently self-improve the constitutive model it generated through self-playing, in the same way AlphaGo Zero (the algorithm that outplayed the world champion in the game of Go) improves its gameplay. Our numerical examples show that this automated meta-modeling framework does not only produces models which outperform existing cohesive models on benchmark traction–separation data, but is also capable of detecting hidden mechanisms among micro-structural features and incorporating them in constitutive models to improve the forward prediction accuracy, both of which are difficult tasks to do manually.

Keywords: Meta-modeling; Traction–separation law; Data-driven computational mechanics; Path-dependent responses; Fracture opening and closure

1. Introduction

Constitutive responses of interfaces are important for a wide spectrum of problems that involve spatial domain with embedded strong discontinuity, such as fracture surfaces [1–4], slip lines [5,6], joints [7] and faults [8–10]. While earlier modeling efforts, in particular those involving the modeling of cohesive zones, often solely focus on mode I kinematics, the mixed mode predictions of traction–separation law relations are critical for numerous applications, ranging from predicting damage upon impacts [11], to predicting seismic events [12]. Park et al. [2] provide a comprehensive account of the major characteristic of traction–separation laws and conclude that, while there are differences in details, most of the traction–separation laws obey a number of universal principles, such as

* Corresponding author.
E-mail address: wsun@columbia.edu (W. Sun).

https://doi.org/10.1016/j.cma.2018.11.026
0045-7825/

the indifference of any superimposed rigid-body motion, the finite work required to create new surface, the existence of characteristic length scales, and the vanishing of cohesive traction with sufficient separations.

In the case where the loading history is not monotonic, constitutive responses of interfaces often become path-dependent. For instance, geomaterials, such as fault gauges, are known to exhibit rate- and state-dependent frictional responses [13–17]. While there are phenomenological models designed to capture the path-dependent responses of the interfaces, a recent trend that gains increasing popularity is to replace the phenomenological traction–separation laws with a computational homogenization procedure to capture the responses of materials with heterogeneous microstructures (cf. Moës et al. [18], Hirschberger et al. [19,20]). Nevertheless, as pointed out previously in Wang and Sun [9], the major issue of applying hierarchical multiscale coupling on interfacial problems is the increasing computational demand due to the large number of required representative elementary simulations, a trade-off that is widely known in FEM^2 [21] and other homogenization-based multiscale methods, such as DEM-FEM [22–30].

To overcome this computational barrier, surrogate models are often derived to replicate the homogenized responses of sub-scale simulations [31–37]. Nevertheless, since surrogate models are often constitutive laws hand-crafted by modelers to incorporate morphology-dependent features [36], deriving, verifying and validating a surrogate model that can incorporate the essential information to yield macroscopic predictions with sufficient accuracy and robustness remain difficult and time-consuming. Data-driven models such as Le et al. [38], Bessa et al. [39], Versino et al. [40], Kafka et al. [41] and Wulfinghoff et al. [42] attempted to overcome this issue via supervised machine learning (e.g. neural network [43], symbolic regression model [40]) and unsupervised machine learning (e.g. dimensional reduction, feature extraction and clustering [39,42]).

Recent work by Wang and Sun [9] attempted to resolve this issue by building a generic recurrent neural network that can easily incorporate different types of sub-scale information (e.g. porosity, fabric tensor, and relative displacement) to predict traction. This technique uses the concept of directed graph on the transfer learning approach (cf. Pan et al. [44]) in which multiple neural networks trained to make predictions on other physical quantities (e.g. relationship between porosity and fabric tensor) are re-used to generate additional inputs for predicting traction. However, the determination of the optimal input information (in addition to the displacement jump history) and configurations of information flow that enhances the prediction accuracy still requires a time-consuming trial-and-error procedure (cf. Section 4.3 Wang and Sun [9]).

In this work, we introduce a general artificial intelligence approach to automate the creation and validation of traction–separation models. Unlike the previous approach in which neural networks are often used to either identify material parameters or create black-box constitutive laws, this work focuses on leveraging the capacity of a computer to improve via self-playing, a technique commonly referred as (deep) reinforcement learning in the computer science community [45–47]. In the past two years, the functionality of algorithms automatically generated from deep reinforcement learning have achieved remarkable success. In many cases, the demonstrated capacities were thought to be impossible in the past. For instance, the algorithm trained by deep reinforcement learning created by a company called DeepMind is able to outperform human experts in Go, Chess and Atari games. The most exciting part of this achievement is that, unlike previous AI such as the IBM Deep Blue, the deep reinforcement learning does not rely on hand-crafted policy evaluation functions and is therefore applicable to different kinds of games once they are defined and implemented.

This success motivates us to propose a meta-modeling approach where deep reinforcement learning may generate constitutive laws for (1) a given set of data, (2) a well-defined objective, and (3) a given set of universal principles. To achieve this goal, we recast the process of writing a constitutive model as a game with components suitable for deep reinforcement learning, involving a sequence of actions completely compatible with the stated rules (i.e., the law of physics). First, we define the model score, which could be any objective function suitable for a given task. For instance, this objective can be minimizing the discrepancy between calibrated experimental results and ***blind predictions*** measured by a norm, or a constrained optimization problem that gives considerations on other attributes such as consistency, speed, and robustness [16]. Once the score (i.e., the objective) is clearly defined, we then implement the rules, which are the universal principles of mechanics, such as material frame indifference, laws of thermodynamics. These rules are applied in an environment in which scores are sampled. In the case of traction–separation law, the environment is simply the validation process itself.

Following this, we then define the action space which consists of a number of actions available for the modelers to write constitutive models. Once the action space and the model score are defined, we leverage the directed graph modeling technique to generate a state. The state at the end of each game represents a constitutive model automatically

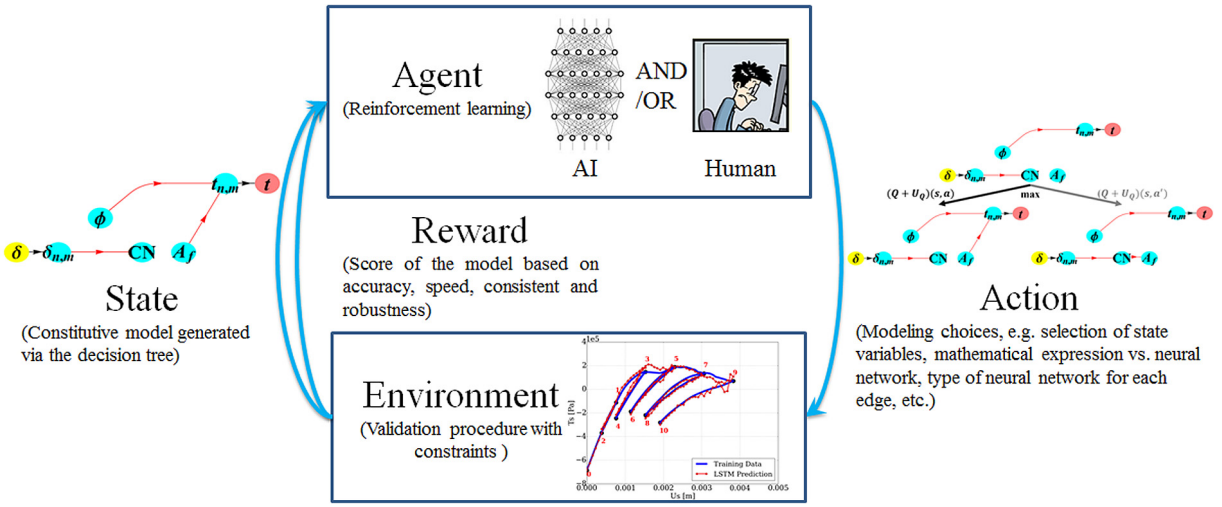


Fig. 1. Scheme of the reinforcement learning algorithm in which an agent interacts with environment and receives rewards. Through exploration, the agent then determines better actions to achieve a particular goal defined by the reward. In our case, the reward is the score which represents the quality of the forward prediction, the action is any possible activities required to derive a constitutive law, and the environment is the procedure that compares the predictions with the benchmark data. Actual graphs of the Environment, State and Action will be detailed in the subsequent sections.

generated from the computer algorithm. In reality, the action space could be of very high dimensions such that manually deriving, implementing, verifying and validating all possible configurations are not feasible. This situation is similar to playing the games of chess and Go where the number of possible combinations of decisions or moves (each can be represented by a decision tree) remains finite but is so enormous that it is not possible to seek the optimal moves by exhausting all possibilities [48].

With the state, action, rule and objective defined, the most critical part is to assign reward for each action. In principle, if the action space is of very low dimension, i.e., there are not many ways to model the physical processes, then the reward for each action can be determined by exhausting all the possible model configurations. However, in the case of writing a complex traction–separation model, we cannot evaluate the quality of the model until its predictions are compared with benchmark data. Therefore, the ability to approximate the reward for each action (in our case the modeling choices) without the need to evaluate all the available options becomes crucial for the success of the meta-modeling approach.

The deep reinforcement learning is therefore ideal for us to achieve this goal. We can approximate the rewards via neural networks and the Bellman expectation equation [49,50]. By repeatedly generating new constitutive laws (i.e., playing the game of writing models), the agent will use the reward obtained from each played game, in analogy with the binary game result (win/loss) at the end of a Go game, to update the action probabilities and value functions to improve the agent's ability to write good constitutive laws. Through sufficient self-plays, the reinforcement learning algorithm then improves the modeling choices it made over time until it is ready for predictions. (See Fig. 1.)

There are a few major upshots for this approach. First, once the reinforcement learning algorithm is established, it can serve as a model generator without any human intervention. Second, since we regard the validation process as the environment component of the reinforcement learning, the performance of a resultant model is simultaneously evaluated and therefore validations are always a part of the model writing process. Third, the meta-modeling approach may easily embed any existing model previously hand-crafted by domain experts into the action space without re-implementing a new model. These unique capabilities enable us to have an unbiased tool to evaluate how well existing models fulfill a particular objective. Furthermore, since the model generation procedure is automated once an objective function is defined, this work may potentially eliminate the need of writing multiple incremental models for the same materials over time. Finally, this modeling approach is particularly powerful for discovering hidden physical coupling mechanisms that are otherwise too subtle to detect with human observation.

The rest of the paper is organized as follows. We first review the directed graph approach that enables us to generate and utilize a decision tree to represent the modeling process (Section 2). The definition of model scores is then

described in Section 3. We then provide a formal definition of a game invented to generate traction–separation laws for predictions (Section 4). This is followed by a description on how to use the reinforcement learning for the traction–separation law generation (Section 5). Two numerical experiments are then used to showcase the performance of the automated meta-modeling approach using synthetic data from microscale discrete element simulations (Section 6). The major findings are then summarized in the conclusions.

2. Representing traction–separation law in directed graph

In this section, we introduce a building block for a simplified and extensible game that generates traction–separation laws by considering the relationships among different types of data collected from sub-scale simulations. In this game, the goal is to find a specific way to link different types of data such that a score function is maximized. Before we introduce the formal definition of the game, one necessary step is to recast the algorithm that leads to predictions from constitutive laws as a network of unidirectional information flow, i.e., a directed graph (also referred to as digraph) [9,51–54]. Recall that a digraph $D = (V, E)$ is an ordered pair of non-empty finite sets which consists of a vertex set V and an edge set E [55]. Each edge connects a source vertex (tail) to a target vertex (head). Following the treatment in Sun [53] and Wang and Sun [9], the following **rules** are applied to generate the traction–separation law.

1. The traction $\boldsymbol{t}$ is placed as the only leaf of the digraph (i.e., the vertex that is not source to any other vertices).
2. The displacement jump $\boldsymbol{\delta}$ is placed as the only root of the digraph (i.e., the vertex that is not target of any other vertices).
3. There may exist isolated vertices in the digraph, i.e., some internal variables or microstructural features between $\boldsymbol{\delta}$ and $\boldsymbol{t}$ may not contribute to the final completed digraph and the corresponding constitutive model.
4. The digraph is acyclic, which means that there must be no cycle in the digraph.
5. If a vertex has sources or targets connected to it, it must be on at least one of the paths leading from $\boldsymbol{\delta}$ to $\boldsymbol{t}$. This ensures that an internal variable, once considered, is fully incorporated into the final constitutive model.

In our previous published work (cf. Sun [53], Wang and Sun [9]), we prescribed theoretical models or, in some cases, neural network models to create linkages and enforce the hierarchy among physical quantities (e.g. porosity–permeability relation). While this treatment is convenient for software engineering and code design [54], this approach only works if we have a prior knowledge about the relationships among the physical quantities. While one may presumably make ad hoc assumptions to complete the models, such a treatment is often at the expense of robustness. Another possible remedy is to gather all the measurement and data one may possibly obtain from observations and experiments, then find the key mechanisms that incorporate the most essential physics (e.g. the critical state plasticity for soil). This latter approach can be re-expressed as a problem in the directed graph in which we only know the elements of the vertex set but have no idea whether and how these vertices are connected, except that the traction is the leaf and the displacement jump is the root of the directed graph. Note that, in reality, the creation of a deterministic constitutive law does not only limit at determining connections among vertices (physical quantities), but also includes finding hidden vertices and appropriate edges. These actions are not modeled in this paper, but will be considered in future studies. Furthermore, while our focus in this paper is on deriving the traction–separation laws, in principle, the idea can be easily extended to other problems, such as the stress–strain relation for bulk materials, the porosity-temperature-fabric-tensor-permeability relations for porous media, among others.

For demonstration purposes, we consider a constitutive law $\boldsymbol{t}(\boldsymbol{\delta}, \boldsymbol{q})$ that predicts the traction vector $\boldsymbol{t}$ based on the history of the displacement jump $\boldsymbol{\delta}$ over a cohesive or cohesive-frictional surface with the normal direction vector being $\boldsymbol{n}$. $\boldsymbol{q}$ is a collection of state variables with n degrees of freedom, i.e., $q_1, q_2, q_3, \ldots, q_n$. We use sub-scale discrete element simulations to generate synthetic data and attempt to create a traction–separation model which can replicate the constitutive responses of complex loading histories.

Imposing restrictions of material frame indifference and assuming isotropic cohesive-frictional surface, the traction–separation model can be simplified to [11]

$$\boldsymbol{t}(\boldsymbol{\delta}, \boldsymbol{q}) = \boldsymbol{t}(\delta_n, \delta_m, \boldsymbol{q}), \tag{1}$$

where $\delta_n = \boldsymbol{\delta} \cdot \boldsymbol{n}$ and $\delta_m = |\boldsymbol{\delta_m}| = |\boldsymbol{\delta} - \delta_n \boldsymbol{n}|$. Hence, the traction $\boldsymbol{t}$ is related to its components t_n and t_m that

$$\boldsymbol{t}(\boldsymbol{\delta}, \boldsymbol{q}) = t_n(\delta_n, \delta_m, \boldsymbol{q})\boldsymbol{n} + t_m(\delta_n, \delta_m, \boldsymbol{q})\frac{\boldsymbol{\delta_m}}{\delta_m}. \tag{2}$$

The internal variables in $\boldsymbol{q}$, if the cohesive surface is composed of a thin layer of granular materials, can be chosen among a large set of geometrical measures on micro-structural attributes [25,28]. In this work, we first manually select the following measures to be the intermediate vertices (the vertices that are neither the leaves nor the roots) to make forward predictions on the traction vector.

- Porosity ϕ, the ratio between the volume of the void and the total volume of a representative volume element (RVE) of the material layer.
- Coordination number $CN = N_{\text{contact}}/N_{\text{particle}}$ where N_{contact} is the number of particle contacts and N_{particle} is the number of particles in the RVE.
- Fabric tensor $\boldsymbol{A}_f = \frac{1}{N_{\text{contact}}} \sum_{c=1}^{N_{\text{contact}}} \boldsymbol{n}^c \otimes \boldsymbol{n}^c$, where $\boldsymbol{n}^c$ is the normal vector of a particle contact c, $c = 1, 2, \ldots, N_{\text{contact}}$ in the RVE.
- Strong fabric tensor $\boldsymbol{A}_{sf} = \frac{1}{N_{\text{strongcontact}}} \sum_{c=1}^{N_{\text{strongcontact}}} \boldsymbol{n}^c \otimes \boldsymbol{n}^c$, where $\boldsymbol{n}^c$ is the normal vector of a strong particle contact (having a compressive normal force greater than mean contact force) c, $c = 1, 2, \ldots, N_{\text{strongcontact}}$ in the RVE.

All particle contacts inside the RVE can form an undirected graph with particles as vertices and interactions as edges. Some quantitative measures of this undirected graph of particle connectivity can be included in the internal variables $\boldsymbol{q}$ as additional microstructural characteristics. Here, we focus on four measures, which are computed using the software package NetworkX (Hagberg et al. [56]), and their detailed explanations can be found in the software documentation.

- d_a, degree assortativity, a scalar value between -1 and 1 measuring the similarity of connections in the graph with respect to the node degree.
- c_t, transitivity coefficient, $c_t = 3\frac{n_{triangles}}{n_{triads}}$, the fraction between the number of triangles and the number of triads present in contact graph.
- l_{sp}, average shortest path length in the contact graph.
- ρ_g, density of the graph, $\rho_g = \frac{2m}{n(n-1)}$, where n is the total number of nodes and m is the total number of edges in the graph.

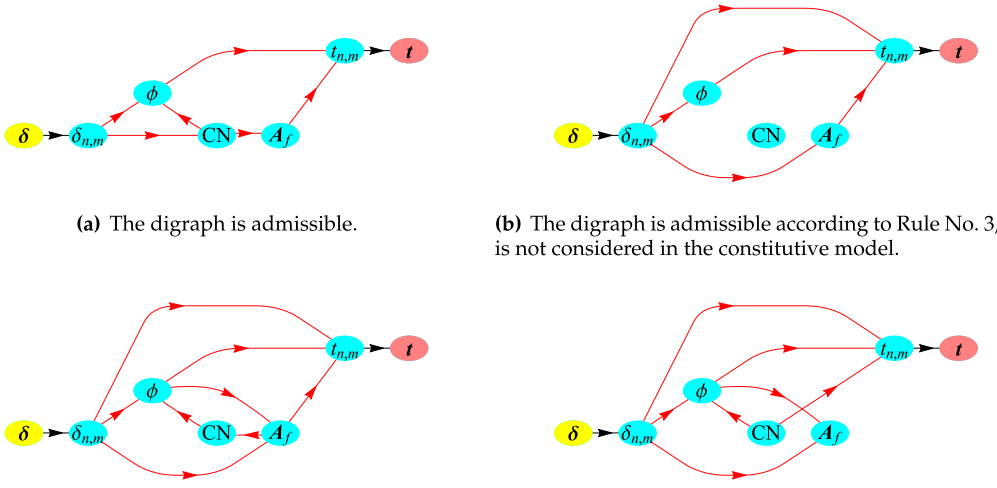
To sum up, in the digraph representations of traction–separation models, $\boldsymbol{\delta}$ is the root and $\boldsymbol{t}$ is the leaf, and currently we consider $\boldsymbol{q}$ to be a subset of the following set of physical quantities $\{\delta_{n,m},\ t_{n,m},\ \phi,\ CN,\ \boldsymbol{A}_f, \boldsymbol{A}_{sf},\ d_a,\ c_t,\ l_{sp},\ \rho_g\}$. For the edges, we classify them as either "definitions" (such as $t_{n,m} \rightarrow \boldsymbol{t}, \boldsymbol{\delta} \rightarrow \delta_{n,m}$) which are determined by universal principles in mechanics and should not be modified, or the "phenomenological relations" (such as $\delta_{n,m} \rightarrow \boldsymbol{A}_f$, $\phi \rightarrow CN$, $l_{sp} \rightarrow t_{n,m}$) which incorporate material parameters chosen to fit experimental data. The latter category of edges provide opportunities for researchers to propose hand-crafted constitutive relations of different degrees of complexities. For example, their forms can be linear, quadratic, exponential functions or be approximated by artificial neural networks (ANNs). For illustration purposes, we consider a simple digraph of traction–separation models involving only the nodes $\{\boldsymbol{\delta},\ \boldsymbol{t},\ \delta_{n,m},\ t_{n,m},\ \phi,\ CN,\ \boldsymbol{A}_f\}$. Fig. 2 provides examples of two admissible and two illegal digraph configurations according to the Rules 1–5.

3. Score system for model evaluation and objective function

A score system must be introduced to evaluate the generated directed graphs for constitutive models such that the accuracy and credibility in replicating the mechanical behavior of real-world materials can be assessed. This score system may also serve as the objective function that defines the rewards for the deep reinforcement learning agent to improve the generated digraphs and resultant constitutive laws. In this work, we define the score as a positive real-valued function of the range [0, 1] which depends on the measures A_i $(i = 1, 2, 3, \ldots, n)$ of n important features of a constitutive model,

$$\text{SCORE} = F(A_1, A_2, A_3, \ldots, A_n), \tag{3}$$

where $0 \leq A_i \leq 1$. Some features are introduced to measure the performance of a model such as the accuracy and computation speed. Other features are introduced to enforce constraints to ensure the admissibility of a constitutive model, such as the frame indifference and the thermodynamic consistency. Suppose there are n_{pfm} measures of



(a) The digraph is admissible.

(b) The digraph is admissible according to Rule No. 3, CN is not considered in the constitutive model.

(c) The digraph violates Rule No. 4, since there exists a cycle $\phi \to \boldsymbol{A}_f \to CN \to \phi$.

(d) The digraph violates Rule No. 5, since CN and $\boldsymbol{A}_f$ are not on any paths leading from $\boldsymbol{\delta}$ to $\boldsymbol{t}$.

Fig. 2. Examples of admissible directed graphs (a–b) and illegal directed graphs (c–d) representing information flow in traction–separation models involving internal physical quantities of porosity ϕ, coordination number CN and fabric tensor $\boldsymbol{A}_f$. The yellow node of separation $\boldsymbol{\delta}$ refers to the root node, the pink node of traction $\boldsymbol{t}$ refers to the leaf node, and the cyan nodes refer to intermediate nodes. The black arrows refer to "definition" or "universal principles" edges. The red arrows refer to "phenomenological relations" edges. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

performance features A_i^{pfm} and n_{crit} measures of critical features A_i^{crit} in the measure system of constitutive models, the score takes the form,

$$\text{SCORE} = \left(\prod_{j=1}^{n_{\text{crit}}} A_j^{\text{crit}}\right) \cdot \left(\sum_{i=1}^{n_{\text{pfm}}} w_i A_i^{\text{pfm}}\right), \tag{4}$$

where $w_i \in [0, 1]$ is the weight associated with the measure A_i^{pfm}, and $\sum_{i=1}^{n_{\text{pfm}}} w_i = 1$. In this section, two examples of measures of accuracy A_{accuracy} and prediction consistency $A_{\text{consistency}}$ are presented.

3.1. Accuracy of calibrations and forward predictions

In this work, the abilities of the models to replicate calibration data and make forward predictions are considered separately. Here we introduce a cross-validation procedure in which the dataset used for training the models (e.g. identifying material parameters (e.g. Wang et al. [16], Liu et al. [36]) or adjusting weights of neurons in recurrent neural networks (e.g. Lefik and Schrefler [43], Wang and Sun [9]) is mutually exclusive to the testing dataset used to evaluate the quality of blind predictions. The details of the generation of these calibration and testing datasets using frictional discrete element simulations are presented in the Appendix. Both calibration and blind prediction results are compared against the target data. The mean squared error (MSE) commonly used in statistics and also as objective function in machine learning is chosen as the error measure for each data sample i in this study, i.e.,

$$\text{MSE}_i = \frac{1}{N_{\text{feature}}} \sum_{j=1}^{N_{\text{feature}}} [\mathcal{S}_j(Y_{i_j}^{\text{data}}) - \mathcal{S}_j(Y_{i_j}^{\text{model}})]^2, \tag{5}$$

where $Y_{i_j}^{\text{data}}$ and $Y_{i_j}^{\text{model}}$ are the values of the jth feature of the ith data sample, from target data value and predictions from constitutive models, respectively. N_{feature} is the number of output features. $\mathcal{S}_j$ is a scaling operator (standardization, min–max scaling, ...) for the output feature $\{Y_{i_j}\}$, $i \in [1, N_{\text{data}}]$.

The empirical cumulative distribution functions (eCDFs) are computed for MSE of the entire dataset $\{\text{MSE}_i\}$, $i \in [1, N_{\text{data}}]$, for MSE of the training dataset $\{\text{MSE}_i\}$, $i \in [1, N_{\text{traindata}}]$ and for MSE of the test dataset $\{\text{MSE}_i\}$, $i \in$

$[1, N_{\text{testdata}}]$, with the eCDF defined as [57],

$$F_N(\text{MSE}) = \begin{cases} 0, & \text{MSE} < \text{MSE}_1, \\ \dfrac{r}{N}, & \text{MSE}_r \le \text{MSE} < \text{MSE}_{r+1},\ r = 1, \ldots, N-1, \\ 1, & \text{MSE}_N \le \text{MSE}, \end{cases} \tag{6}$$

where $N = N_{\text{data}}$, or $N_{\text{traindata}}$, or N_{testdata}, and all $\{\text{MSE}_i\}$ are arranged in increasing order. A measure of accuracy is proposed based on the above statistics,

$$A_{\text{accuracy}} = \max\left(\frac{\log[\max(\varepsilon_{P\%}, \varepsilon_{\text{crit}})]}{\log \varepsilon_{\text{crit}}}, 0\right), \tag{7}$$

where $\varepsilon_{P\%}$ is the Pth percentile (the MSE value corresponding to $P\%$ in the eCDF plot) of the eCDF on the entire, training or test dataset. $\varepsilon_{\text{crit}} \ll 1$ is the critical MSE chosen by users such that a model can be considered as "satisfactorily accurate" when $\varepsilon_{P\%} \le \varepsilon_{\text{crit}}$.

3.2. Consistency of accuracy between calibrations and forward predictions

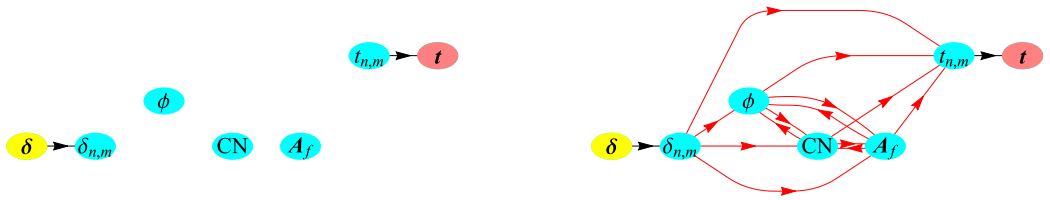
For the examination of the consistency in model predictions on training data and test data, the K-sample Anderson–Darling (AD) test of goodness-of-fit (gof) is conducted to check whether the eCDFs of training and test data come from the same probability distribution, while this distribution is unspecified [58,59]. It is a non-parametric hypothesis test and determines whether the null hypothesis H_0 that the two eCDFs come from the same continuous distribution can be rejected or not, under a chosen significance level α_{gof}. The method consists of calculating a normalized AD test statistic, critical values of the AD statistic that depends on the sample sizes, and a p-value indicating the approximated significance level at which H_0 can be rejected. If the p-value is smaller than the significance level α_{gof}, the H_0 hypothesis is rejected. Otherwise there is insufficient evidence to reject H_0. In this work, we define the following binary measure for the consistency of the MSE distributions, with the significance level α_{gof},

$$A_{\text{consistency}} = H^{\alpha_{\text{gof}}} = \begin{cases} 0, & p\text{-value} < \alpha_{\text{gof}}, \\ 1, & p\text{-value} \ge \alpha_{\text{gof}}. \end{cases} \tag{8}$$

4. Game of the traction–separation law

Our focus in this paper is primarily on the meta-modeling game invented for generating traction–separation models. Nevertheless, similar games can be defined for generating other types of constitutive models based on the ideas presented in this work. With the directed graph representations of traction–separation models as presented in Section 2, the process of developing a model can be recast as a game of making a sequence of decisions in generating edges between nodes in the digraphs. The player of the game can be a human or an AI agent. The game starts with an initial "game board" of digraph with predefined nodes and predefined "definition" edges (e.g. displacement and deformation gradient, displacement jump vector and the scalar components in a given basis), while no "phenomenological relation" edge is formed among them. Each step of the game consists of activating only one edge among all possible choices of edges in the predefined action space, following the predefined rules of the game. The game terminates when a complete and admissible digraph following the rules in Section 2 is established. The output models of the game are measured by a score system as presented in Section 3.

The game can be mathematically formalized as a Markov decision process. The human or AI agent observes the state of the game s_t at the current step t from the game environment (the directed graph that represents a constitutive model) in the form of a collection of binaries indicating the on/off status of each valid edge choices in the action space. The agent takes an action a_t on the game environment in the form of an integer indicating the next edge to switch on in the action space. The action a_t is sampled from a vector of probabilities $\pi(s_t)$ of taking each valid action from the state s_t. Consequently, the state of the game becomes s_{t+1} at the next step $t + 1$. The agent also receives a reward r_{t+1} for the action a_t of taking the game state from s_t to s_{t+1}. Each policy applied in a complete gameplay produces a particular trajectory $s_0, a_0, r_1, s_1, a_1, r_2, \ldots, a_{t-1}, r_t, s_t, a_t, \ldots, a_{T-1}, r_T, s_T$. Once a complete constitutive model is generated, the model score is evaluated. The final reward r_T is defined as follows: if the current score is higher than the average score of models from a group of already played games by the agent, then the current model wins and $r_T = 1$, otherwise, the current model loses and $r_T = -1$. The average score can be initialized to 0 for the first game.



(a) Initial configuration of the "game board" **(b)** All possible actions on the "game board"

Fig. 3. A game of traction–separation model for the digraph example in Fig. 2. (a) The initial "board" on which the game is played. (b) All possible actions for picking the edges connecting the nodes are represented by the red arrows. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 1
Key ingredients of the game of constitutive models in directed graph.

Environment	Benchmark training and test data, idealized multigraph for constitutive models
Agent	Human or AI
State s	A list of binaries indicating the on/off status of each valid edge choice
Action a	An integer indicating the next edge to switch on from the current game state
Reward r	Win (1) / loss (-1) according to the score of the constitutive model in Section 3
$\pi(s,a)$	Probability of taking action a at state s
$v(s)$	Expected reward of state s
Q-value $Q(s,a)$	Expected reward from taking action a at state s

Note that the exact values of the rewards $r_{t\leq T}$ are only known at the end of the game, similar to the game of Chess and Go. r_T is determined according to the final score of the generated model. If $r_T = 1$, then all previous intermediate rewards $r_{t<T} = 1$. If $r_T = -1$, then $r_{t<T} = -1$. Before the game reaches an end, however, $r_{t<T}$ can only be estimated by the agent based on its knowledge of the game. For a human agent, both rewards $r_{t<T}$ and move probabilities $\pi(s)$ come from "intuition" gained during many constitutive modeling practices. An experienced modeler estimates the rewards and probabilities more accurately and hence more likely generates better constitutive models. For an AI agent, $r_{t<T}$ and $\pi(s)$ can either be approximated by hand-crafted mathematical functions or neural networks, as demonstrated in the deep reinforcement Q-learning in AlphaGo and AlphaGo Zero. These rewards and probabilities are estimated based on the expected game reward of taking action a from state s (Q-value) $Q(s,a)$ and the value of current state $v(s)$. The above-mentioned important quantities for mathematical descriptions of the gameplays are summarized in Table 1. Moreover, the constitutive modeling game is compared side-by-side with the game of Chess, a game more familiar to the public, in Table 2, in the aspects of the board to play on, the permitted actions to execute, the criteria for wining the game, etc.

For illustration purposes, we provide a simple game example for the digraph presented in Fig. 2, which only involves the nodes $\{\boldsymbol{\delta},\ \boldsymbol{t},\ \delta_{n,m},\ t_{n,m},\ \phi,\ CN,\ \boldsymbol{A}_f\}$. Fig. 3 presents the "initial game board" and all possible edges choices in the current game definition. The configuration of the digraph, or the state of the game, can be totally described by a list of binaries for 13 edges $[\delta_{n,m} \to \phi,\ \delta_{n,m} \to CN,\ \delta_{n,m} \to \boldsymbol{A}_f,\ \delta_{n,m} \to t_{n,m},\ \phi \to CN,\ \phi \to \boldsymbol{A}_f,\ \phi \to t_{n,m},\ CN \to \phi,\ CN \to \boldsymbol{A}_f,\ CN \to t_{n,m},\ \boldsymbol{A}_f \to \phi,\ \boldsymbol{A}_f \to CN,\ \boldsymbol{A}_f \to t_{n,m}]$ (The edges $\boldsymbol{\delta} \to \delta_{n,m}$ and $t_{n,m} \to \boldsymbol{t}$ are definitions and always active). The list also represents the entire action space. The action a is an integer $\in [0, 12]$ indicating the next edge ID to activate in the list. The legal moves at the current game state are represented by a list of 13 binaries indicating whether the corresponding edges are allowed to be activated for the next action step. The rules of the legal moves are as follows: (1) if one edge has already been selected, it is excluded from the selection of actions; (2) if an edge between two intermediate nodes has been selected, the other edge involving these two nodes but with opposite direction is also excluded (e.g., The edges $\phi \to CN$ and $CN \to \phi$ are mutually exclusive); (3) the final digraph of a complete traction–separation model must obey the rules in Section 2. This rule is explicitly checked for each action at each game state. All the actions resulting in a final digraph violating the rules in Section 2 will be forbidden. For example, if the edges $\phi \to CN$ and $CN \to \boldsymbol{A}_f$ have already been selected in the current incomplete digraph, then the action to select the edge $\boldsymbol{A}_f \to \phi$ is an illegal move, since it will lead to a final digraph that has a cycle.

Table 2
Comparison of the essential definitions between the game of Chess and the game of constitutive modeling in directed graph.

	Game of Chess	Game of constitutive modeling in directed graph
Definition of game	Make a sequence of decisions to maximize the probability to win	Make a sequence of decisions to maximize the score of the constitutive model
Game board	8×8 grid	Directed graph with predefined nodes of physical quantities and edges of definition or universal principles
Game state	Configuration of chess pieces on the board	Configuration of directed graph representing the constitutive model
Game action	Move chess pieces	Select among modeling choices. For instance 1. What physical quantities are included? 2. How physical quantities are linked? 3. What are the edges between physical quantities?
Game rule	Restrictions on chess piece movements	Universal principles Rules in Section 2 Specific restrictions on edge choices
Game reward	Win, draw or loss (discontinuous)	Win or loss (discontinuous) from comparison of model scores (continuous)
Reward evaluation	Only available at the end	Only available at the end

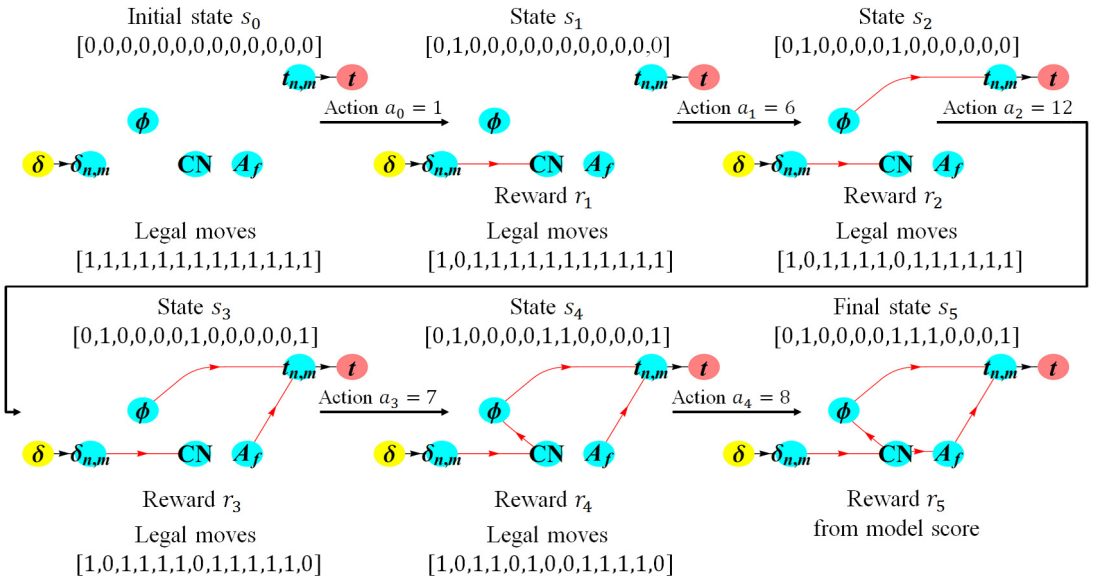


Fig. 4. A gameplay example formalized as a Markov decision process (s_0, a_0, r_1, s_1, a_1, r_2, s_2, a_2, r_3, s_3, a_3, r_4, s_4, a_4, r_5, s_5) for the digraph game in Fig. 3. The states are lists of binaries for 13 edges [$\delta_{n,m} \to \phi$, $\delta_{n,m} \to CN$, $\delta_{n,m} \to \boldsymbol{A}_f$, $\delta_{n,m} \to t_{n,m}$, $\phi \to CN$, $\phi \to \boldsymbol{A}_f$, $\phi \to t_{n,m}$, $CN \to \phi$, $CN \to \boldsymbol{A}_f$, $CN \to t_{n,m}$, $\boldsymbol{A}_f \to \phi$, $\boldsymbol{A}_f \to CN$, $\boldsymbol{A}_f \to t_{n,m}$]. The actions a are integers $\in [0, 12]$ (the list indices start from 0) indicating the next edge ID to activate. The legal moves are lists of binaries indicating whether the edges are allowed to be activated next. Final reward r_5 is determined by the model score evaluated at the end of the game. r_{1-4} are only estimated by "intuitions" on whether the current policy can lead to a win or not, until r_5 is known. Note that the Markov decision process leading to the final digraph configuration s_5 is not unique.

Fig. 4 provides a gameplay example of the constitutive modeling game in Fig. 3, with the evolving mathematical representations of game states, actions and legal actions, as well as the Markov decision process.

The score evaluation (Section 3) requires model calibration on training data, and forward predictions on test data. The procedure for score evaluation is as follows. Once the final digraph configuration is determined, all paths (information flows) leading from $\boldsymbol{\delta}$ to $\boldsymbol{t}$ and all predecessors for each node in the paths are identified using the graph theory (software package NetworkX). Then, the predecessor nodes for the terminal node $\boldsymbol{t}$ within these paths are identified. Recursively going upstream along all the information flows, the predecessors for these nodes are identified,

until the final predecessor node is the start node $\boldsymbol{\delta}$ only. All the predecessor–successor node pairs can be connected by either mathematical equations frequently used in handcrafted constitutive models (linear, quadratic, exponential, power law, etc.) or artificial neural networks. In this work, we take the advantage of the flexibility of ANNs that they are universal function approximators to continuous functions of various complexity on compact subsets of R^n (Universal approximation theorem, [60]). Moreover, a special type of ANN, recurrent neural network (e.g., long short-term memory LSTM [61], gated recurrent units GRU [62,63]), is used to capture the function of a time series of inputs, which is ideal for replicating the path-dependent material behaviors. In this work, we only use ANN edges to connect nodes, without losing generality of the meta-modeling games. The hybridized constitutive models with both mathematical equation edges and ANN edges will be studied in a separate research. The predecessor–successor node pairs are also inputs and outputs of all ANNs involved in the constitutive model. For example, there are two paths in the final digraph s_5 in Fig. 4: $\{\boldsymbol{\delta} \rightarrow \delta_{n,m} \rightarrow CN \rightarrow \boldsymbol{A}_f \rightarrow t_{n,m} \rightarrow \boldsymbol{t}\}$ and $\{\boldsymbol{\delta} \rightarrow \delta_{n,m} \rightarrow CN \rightarrow \phi \rightarrow t_{n,m} \rightarrow \boldsymbol{t}\}$. Then the three required ANNs are, represented as input–output pairs, $[\delta_{n,m} \rightarrow CN]$, $[CN \rightarrow \phi, \boldsymbol{A}_f]$ and $[\phi, \boldsymbol{A}_f \rightarrow t_{n,m}]$. The parameters in each ANN are calibrated with training data of the input and output features using back propagations. The final output of $\boldsymbol{t}$ is predicted by executing consecutively the ANNs following the established paths from $\boldsymbol{\delta}$ to $\boldsymbol{t}$ in the directed graph. In the numerical examples of this paper, the same neural network architecture is used for all ANNs for all edges in the directed graph: two hidden layers of 32 GRU neurons in each layer, and the output layer is a dense layer with linear activation function. All input and output data are pre-processed by standard scaling using mean values and standard deviations [64]. Each input feature considers its current value and 19 history values prior to the current loading step. Each ANN is trained for 1000 epochs using the Adam optimization algorithm [65], with a batch size of 256.

This particular ANN configuration is chosen after a number of trial-and-error numerical experiments similar to the one described in greater details in Wang and Sun [9]. These experiments shown that the performance of the models is not particularly sensitive to small changes of the design parameters we considered (e.g. number of layers, number of neurons per layer), provided that a recurrent network is used and successfully trained. In fact, we found that the prediction accuracy does not exhibit significant different while we replace the long short-term memory neurons with the gated recurrent neurons. This robustness ensures that the predictions of each edges are sufficiently accurate and therefore potentially leads to more accurate predictions from the entire directed graph and ultimately higher game rewards.

In principle, other designs of neural networks can also be used as the universal function approximators for the digraph edges, provided that the true path-dependent relations are accurately captured. While it is possible that there exists better RNN setups, determining this optimal setup from all possible combinations of choices manually will nevertheless require a significant amount of trial-and-error effort. One way of overcoming this obstacle is to use the reinforcement learning to automate this trial-and-error procedure which determines the optimal setup for each edge in the directed graph. Exploring this fine-tuning option is out of the scope of this study, but we will nevertheless investigate this further in future studies.

5. Deep reinforcement learning for generating constitutive laws

With the game of constitutive modeling completely defined, a deep reinforcement learning (DRL) algorithm is employed as a guidance of taking actions in the game to maximize the final model score (Fig. 5). This tactic is considered one of the key ideas leading to the major breakthrough in AI playing the game of Go (AlphaGo Zero) [47], Chess and shogi (Alpha Zero) [66] and many other games. The learning is completely free of human interventions. It does not need previous human knowledge in traction–separation model as a starter database. The AI agent simply learns to improve from a number of games it played and from the corresponding model scores and game rewards, even if the initially generated digraph configurations make very little sense for a traction–separation model. Moreover, during the self-plays and training, no human guidance is needed.

A (deep) neural network f_θ with parameters θ (weights, bias, ... of the artificial neurons) takes in the current configuration of the directed graph of the constitutive law s and outputs a policy vector $\boldsymbol{p}$ with each component $p_a = p(s, a)$ representing the probability of taking the action a from state s, as well as a scaler v estimating the expected score of the constitutive law game from state s, i.e.,

$$(\boldsymbol{p}, v) = f_\theta(s). \tag{9}$$

These outputs from the policy/value network guide the game play from the AI agent.

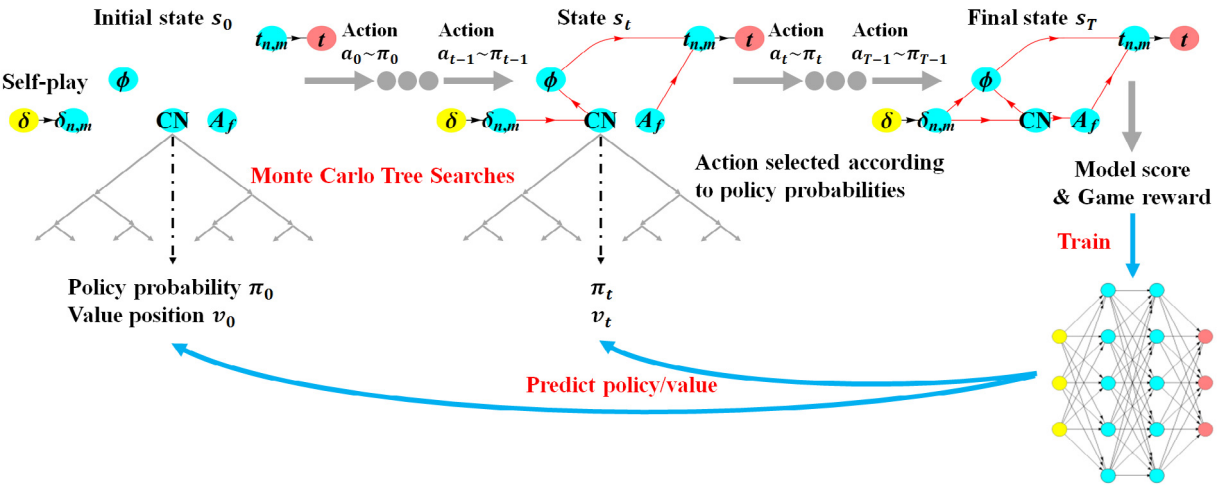


Fig. 5. Self-play reinforcement learning of traction–separation law.

At each state s, the action to take is sampled from an action probability $\boldsymbol{\pi}(s)$. This probability is based on the policy $\boldsymbol{p}$ predicted from the neural network enhanced by a Monte Carlo Tree Search (MCTS) [67]. The search tree is composed of nodes representing states s of the game, and edges representing permitted actions a from s. Each edge (s, a) possesses a list of statistics $[N(s, a), W(s, a), Q(s, a)]$, where $N(s, a)$ is the number of visits to the edge during MCTS search, $W(s, a)$ is the total action value and $Q(s, a) = \frac{W(s,a)}{N(s,a)}$ is the mean action value. The search procedure consists of firstly a recursive selection of a sequence of optimal actions $a^0, a^1, a^2, \ldots$ leading to the corresponding child states $s^1, s^2, s^3, \ldots$, starting from the root state s^0, until a leaf node of state s^l (that has never been encountered before in the search) is reached. The criterion for selecting a new action from a state s is that this action a maximizes the upper confidence bound $U(s, a)$ of the Q-value, among all valid actions. The upper bound is defined as

$$U(s,a) = Q(s,a) + U_Q(s,a) = Q(s,a) + c_{puct}\, p(s,a) \frac{\sqrt{\sum_b N(s,b)}}{1 + N(s,a)}. \tag{10}$$

where c_{puct} is a parameter controlling the level of exploration. If s^l is not a terminal state that ends the game, then its $\boldsymbol{p}(s^l)$ and $v(s^l)$ are predicted from the policy/value neural network $f_\theta(s^l)$. The search tree is expanded and the statistics for each edge (s^l, a) is initialized to $[N(s, a) = 0, W(s, a) = 0, Q(s, a) = 0]$. Otherwise, $v(s^l)$ is equal to the final reward of the constitutive modeling game. Finally, $v(s^l)$ is propagated back to the parent states $\{s^0, s^1, s^2, \ldots, s^l\}$ and actions $\{a^0, a^1, a^2, \ldots, a^{(l-1)}\}$ traversed during the search. Their statistics are updated as

$$N(s^i, a^i) = N(s^i, a^i) + 1,\ W(s^i, a^i) = W(s^i, a^i) + v(s^l),\ Q(s^i, a^i) = \frac{W(s^i, a^i)}{N(s^i, a^i)},\ \text{for all } i < l. \tag{11}$$

The MCTS procedure is repeated a number of times. The searches in MCTS eventually yield a vector of search probabilities $\boldsymbol{\pi}(s^0)$ recommending actions to take from the root position s^0. $\boldsymbol{\pi}(s^0)$ is proportional to the exponentiated visit count for each edge, i.e.,

$$\boldsymbol{\pi}(s^0, a) = \frac{N(s^0, a)^{-\tau}}{\sum_b N(s^0, b)^{-\tau}}, \tag{12}$$

where τ is a positive temperature parameter that also controls the level of exploration. The MCTS algorithm for the game of constitutive models is illustrated in Fig. 6.

During one episode of self-play by the AI agent, the above MCTS algorithm is executed for each state s_t in the sequence of encountered states $\{s_0, s_1, s_2, \ldots, s_{T-1}\}$. The root node s^0 of the search tree is set to s_t as the game progresses to the state s_t. All child nodes and their statistics constructed in the MCTS for the prior game states are preserved. The training data for the neural network consists of $(s_t, \boldsymbol{\pi}_t, z_t)$ obtained from a number of full plays of the constitutive law game guided by the aforementioned reinforcement learning algorithm. $\boldsymbol{\pi}_t$ is the estimation of policy

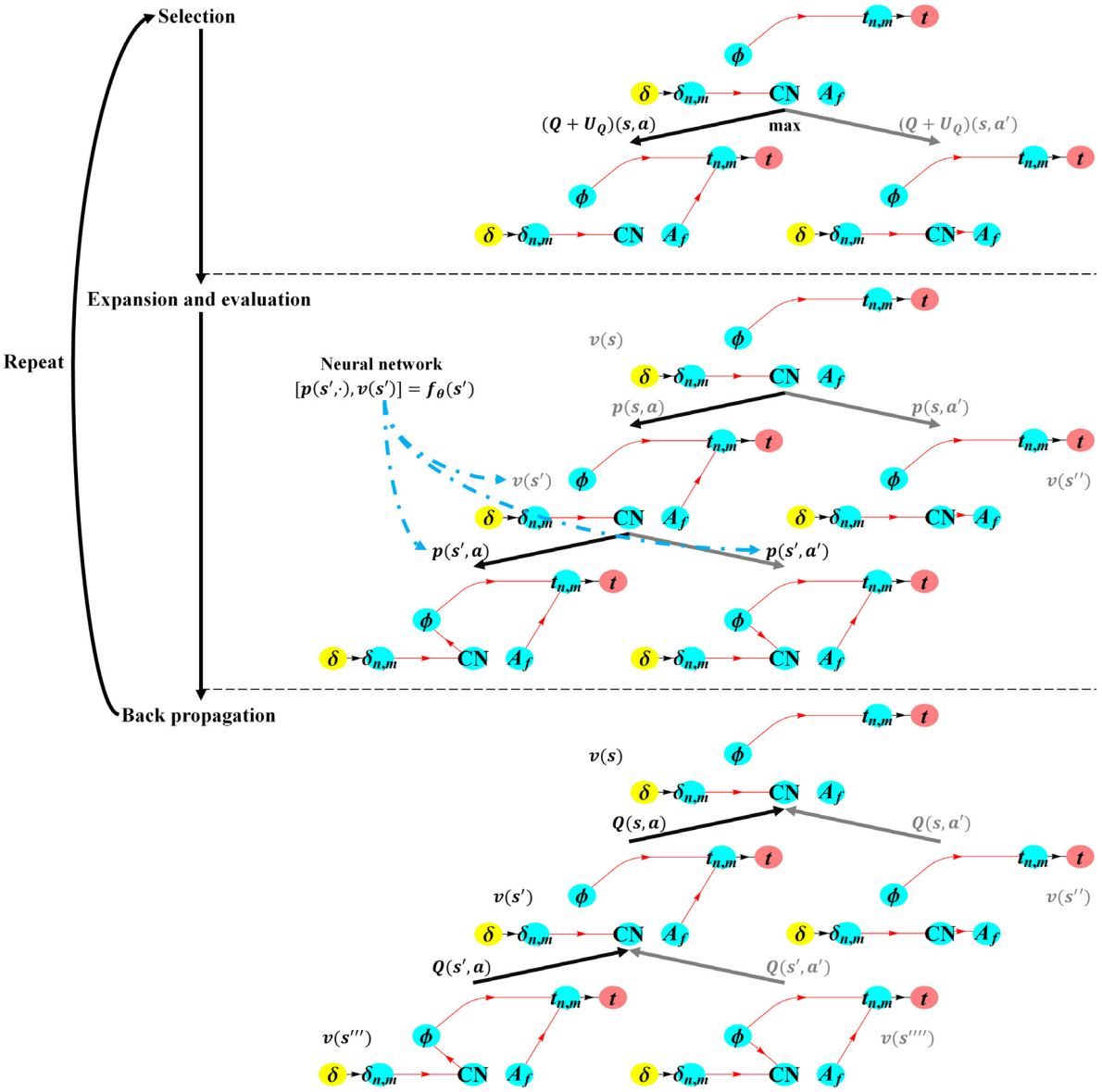


Fig. 6. Monte Carlo Tree Search (MCTS) in a game of constitutive models (figure design adopted from [47]). A sequence of actions are selected from the root state s^0, each maximizing the upper confidence bound $Q(s,a) + U_Q(s,a)$. The leaf node s^l is expanded and its policy probabilities and position value are evaluated from the neural network $(\boldsymbol{p}(s^l), v(s^l)) = f_\theta(s^l)$. The action values Q in the tree are updated from the evaluation of the leaf node. Finally the search probability $\boldsymbol{\pi}(s^0)$ for the root state s^0 is returned to guide the next action in self-play.

after performing MCTS from state s_t and z_t is the reward of the generated constitutive model at the end of the game s_T. The loss function to be minimized by adjusting parameters θ using back propagation is,

$$1 = \sum_t (v(s_t) - z_t)^2 + \sum_t \boldsymbol{\pi}_t \log[\boldsymbol{p}(s_t)], \tag{13}$$

which is the combination of mean squared errors in game reward (1 or -1) and cross-entropy losses in policy probabilities. Hence, accordingly, the activation functions for the output layer are the hyperbolic tangent function $\tanh(x) = \frac{e^{2x}-1}{e^{2x}+1}$ and the softmax function [68]. The procedure of DRL guided self-plays and the following training of

the network f_θ is iterated until the score of the generated directed graph of the constitutive model does not improve further.

The pseudocode of the reinforcement learning algorithm designed to play the meta-modeling game and improves via self-play is presented in Algorithm 1. As demonstrated in Algorithm 1, each complete DRL procedure involves $numIters$ number of training iterations and one final iteration for generating the final converged digraph model. Each iteration involves $numEpisodes$ number of game episodes that construct the training example set $trainExamples$ for the training of the policy/value network f_θ. For decision makings in each game episode, the action probabilities are estimated from $numMCTSSims$ times of MCTS simulations.

Algorithm 1 Self-play reinforcement learning of the meta-modeling game

Require: The definition of meta-modeling game: game environment, state, action, model score, reward, game rules (Section 4).

Randomly initialize the policy/value network f_θ.
Initialize empty set of the training examples $trainExamples \leftarrow []$.
for i in [0,..., $numIters$ (number of training iterations)-1] **do**
    for j in [1,..., $numEpisodes$ (number of game episodes)] **do**
        Initialize the starting game state s.
        Initialize empty tree of the Monte Carlo Tree search (MCTS), set the temperature parameter $\tau = 1$ for "exploration and exploitation".
        while True **do**
            Check for all legal actions at current state s according to the game rules.
            Get the action probabilities $\pi(s, \cdot)$ for all legal actions by performing $numMCTSSims$ times of MCTS simulations.
            Sample action a from the probabilities $\pi(s, \cdot)$
            Modify the current game state to a new state s by taking the action a.
            if s is the end state of a game **then**
                Evaluate the score of the constructed digraph.
                Evaluate the reward r of this game episode according to the model score.
                Break.
        Append the history in this game episode $[s, a, \pi(s, \cdot), r]$ to $trainExamples$.
    Train the policy/value network f_θ with $trainExamples$.
Use the final trained network f_θ and set $\tau = 0.01$ in MCTS for one more iteration of "competitive gameplays" ($numEpisodes$ games) to generate the final converged digraph model.
Exit

This design of algorithm is very similar to the one used in AlphaGo Zero to play and discover knowledge from the Game of Go and chess (cf. Silver et al. [47], Silver et al. [66]). The major difference is that we are now applying the reinforcement learning on a newly design game in which discoveries of physical relationships and quality of blind predictions are all based on making the right decisions in the step-by-step construction of directed graph rather than the movement of the game piece. This similarity of the algorithm design is encouraging, as it demonstrate that the DRL algorithm is closer to reach the goal of becoming a form of general artificial intelligence and that it can be easily applied to many other decision making "games" designed for different research fields [69,70].

6. Numerical experiments and applications

In this section, we present two traction–separation modeling games with different digraph complexities to demonstrate the intelligence, robustness and efficiency of the deep reinforcement learning algorithm on improving the accuracy and consistency of the generated traction–separation models through self-plays. For both examples in Sections 6.1 and 6.2, sub-scale discrete element simulations (DEM) are used to generate synthetic benchmark data for model calibrations and blind prediction evaluations. For brevity, the procedure of database generation from a pre-consolidated representative volume element (RVE) of a frictional contact material is described in the Appendix. Wang and Sun [9] provide a detailed account on the procedures for synthetic data generation, training and testing of ANN

models. In both examples, the accuracy scores of the DRL-generated models are evaluated by material point tests with deformation histories identical to those used to generate the benchmark data. The third numerical example presents a multiscale finite element simulation where the previously auto-generated optimal model is used as a scale-bridging constitutive model for pre-existing interface.

6.1. Numerical Experiment 1: Determining optimal physical relationships for traction–separation laws

In the first example, our goal is to test the DRL algorithm and see whether it can determine the optimal topological relations among microstructural physical quantities of porosity ϕ, coordination number CN and fabric tensor $\boldsymbol{A}_f$. In Wang and Sun [9], the authors use domain expertise, i.e., knowledge from previous literature on fabric tensor and critical state theory to deduce that the porosity and fabric tensor can be used as state variables to improve the forward prediction accuracy of the traction–separation law (cf. Fu and Dafalias [71], Li and Dafalias [72], Sun [14], Wang and Sun [29]). In this work, we do not make any assumption or introduce any interpretation to the meta-modeling computer agent. Instead, we simply make a number of physical quantities measured from discrete element simulations available as vertices in the directed graph but do not introduce any relation (edge) manually. In other words, the edge set that represents the relations of the physical quantities is self-discovered by the computer agent from the reinforcement learning without any human intervention. We document our training procedure and analyze the performance of the models generated by the meta-modeling approach.

The directed graphs, states, actions, rewards and game rules of the modeling game have been defined in Sections 2 and 4, and illustrated in Figs. 2–4. The action space is of dimension 13. Through exhaustive plays of the game, the authors count 3200 possible game states, among which 591 states represent complete and admissible directed graph configurations according to the game rules. The model score is defined as:

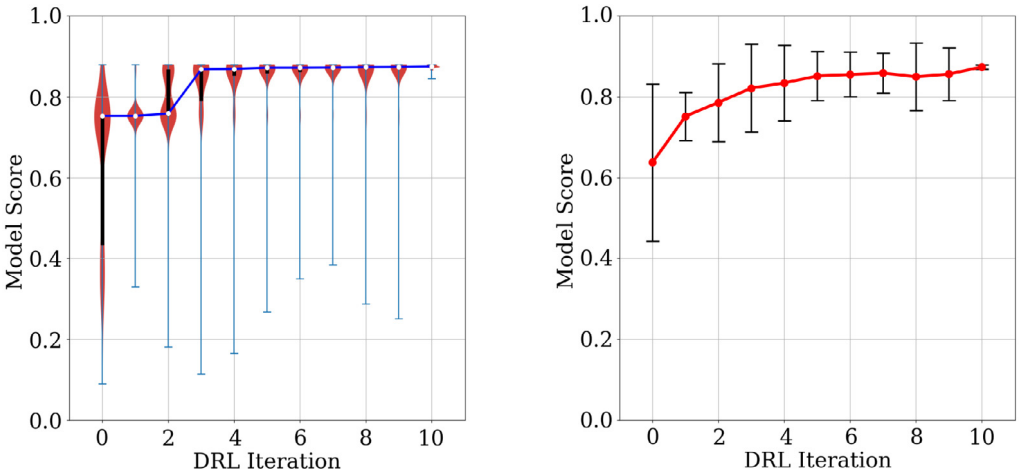
$$\text{SCORE} = 0.45 * A^{\text{calibration}}_{\text{accuracy}} + 0.45 * A^{\text{prediction}}_{\text{accuracy}} + 0.1 * A_{\text{consistency}}, \tag{14}$$

where $P\% = 90\%$ and $\varepsilon_{\text{crit}} = 1e^{-6}$ for accuracy evaluations and $\alpha_{\text{gof}} = 1\%$ for consistency evaluations. The training data for model calibration contains 50 loading cases, and the test data for forward prediction evaluation contains 150 loading cases.

The DRL meta-modeling procedure (Algorithm 1) contains $numIters = 10$ training iterations of "exploration and exploitation" of game strategies, by setting the temperature parameter τ to 1. Then an iteration of "competitive gameplay" ($\tau = 0.01$) is conducted to showcase the performance of the final trained AI agent. Each iteration consists of $numEpisodes = 20$ self-play episodes of the game. Hence one execution of the entire DRL procedure contains $numIters * numEpisodes = 10 * 20 = 200$ game episodes for training the policy/value neural network. Each game starts with a randomly initialized neural network for the policy/value predictions, and each play step requires $numMCTSSims = 20$ MCTS simulations. Then the play steps and corresponding final game rewards are appended to the set of training examples for the training of the policy/value network.

Because each run of the DRL procedure (Algorithm 1) is initialized randomly and the MCTS simulations also involve probability of the action possibilities, the gameplays in each individual DRL procedures could, in theory, lead to different resultant directed graph model for the same set of data, especially if the exploration is not sufficient. To analyze how much influence does the initial randomly directed graph affects the outcome of the gameplay, we conduct a numerical experiment in which 20 independent DRL procedures, each with a different initial configuration, are ran. We then compare the results obtained from these DRL procedures to assess the statistical performance of the DRL and also check whether they all converged to the same directed graph. The gameplay results are presented in Fig. 7. The number of gameplays required for each DRL algorithm to successfully generate the optimal digraph model is 200, which is about 33% of total 591 possible digraphs the brutal-force approach required to evaluate and rank such that the optimal graph can be determined. Yet, these 20 independent DRL procedures with different initial configurations all lead to the same optimal digraph. This optimal digraph is identical to the one determined from the brutal-force approach in our benchmark study.

At the first DRL iteration, the AI agent only knows the rule of the game without any human knowledge on which physical quantities are essential in predicting the traction and how they should be connected. The AI just plays with trial-and-error following strategies guided by random initial neural network and MCTS. This lack of gameplay knowledge can be seen from the widely spread density distribution of model scores between maximum and minimum scores, large interquartile range between 25% and 75%, and the large standard deviation (Fig. 7). In the



(a) Violin plots of the density distribution of model scores in each DRL iteration

(b) Mean value and ± standard deviation of model score in each DRL iteration

Fig. 7. Statistics of the model scores in deep reinforcement learning iterations from 20 separate runs of the DRL procedure for Numerical Experiment 1. Each DRL procedure contains ten iterations 0–9 of "exploration and exploitation" (by setting the temperature parameter $\tau = 1.0$) and a final iteration 10 of "competitive gameplay" ($\tau = 0.01$). Each iteration consists of 20 games. (a) Violin Plot of model scores played in each DRL iteration. The shade area represents the density distribution of scores. The white point represents the median. The thick black bar represents the interquartile range between 25% quantile and 75% quantile. The maximum and minimum scores played in each iteration are marked by horizontal lines. (b) Mean model score in each iteration and the error bars mark ± standard deviation.

subsequent iterations, the AI plays with increasing knowledge of the meta-modeling game reinforced by the ultimate game rewards, and it shows intelligence in keep playing games with better outcomes. This is shown by the increase in median and average of scores, the narrowing of interquartile range and the migration of the density distribution towards higher scores. The automatic learning is very efficient. Statistically, after 5 iterations (100 games out of the total 591 possible game outcomes), the scores already concentrate around the maximum. Few bad games could be played, since the AI is still allowed to explore different game possibilities to avoid convergence to local maximum. The strength of the AI agent after 10 iterations is tested by suppressing the "exploration plays", and the outcome game scores show outstanding performances. Fig. 8 illustrates the improvement of knowledge of traction–separation constitutive modeling by four representative digraph games played during the DRL iterations. The traction predictions from the resultant constitutive models are compared against both training data and unseen test data. In addition, four examples of blind predictions from the optimal digraph configuration (the 4th digraph in Fig. 8) obtained in this game are shown in Fig. 9.

6.2. Numerical Experiment 2: Data-driven discovery for enhancement of traction–separation laws

In the second example, we consider another common scenario in which we attempt to convert qualitative observations into quantitative predictions with the help of the reinforcement learning algorithm as a tool for augmented intelligence. The need to interpret observations of mechanisms into predictions is one of the oldest problems in constitutive modeling [73]. For instance, the observation that yielding depends on the amount of normal traction leads to the Mohr–Coulomb yield criterion [74]. The evolution of fabric tensor has been incorporated into the hardening law and the plastic flow rule to capture the induced anisotropy and critical state of sand [75–77]. However, recent advancements on the application of graph theory as well as the experimental techniques such as micro-CT imaging have revealed many geometric measures on the grain connectivity that help explain the onset of shear band [78,79], coherent vortex structure [80] and post-bifurcation behaviors in granular materials [25,26,81]. While these discoveries of new knowledge are indeed encouraging, one cannot make use of them without investing significant efforts and time to derive, verify, and validate new constitutive laws that incorporate new information. Hence the graph-theoretical approaches, although have found great promises on analyzing the granular assemblies obtained from real or virtual

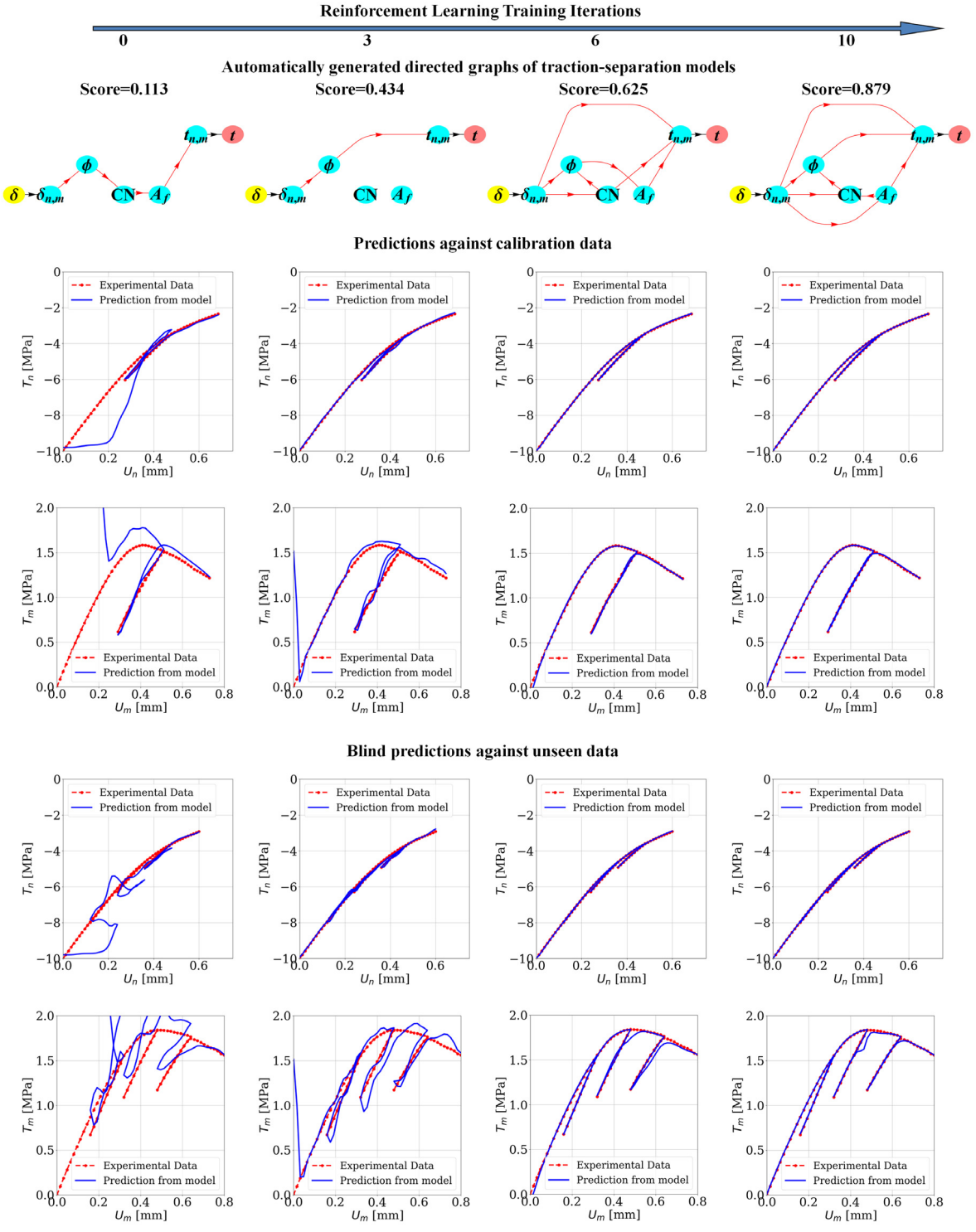


Fig. 8. Knowledge of directed graphs of traction–separation models learned by deep reinforcement learning in Numerical Experiment 1. Four representative digraph games played during the DRL iterations and their prediction accuracy against training and test data are presented.

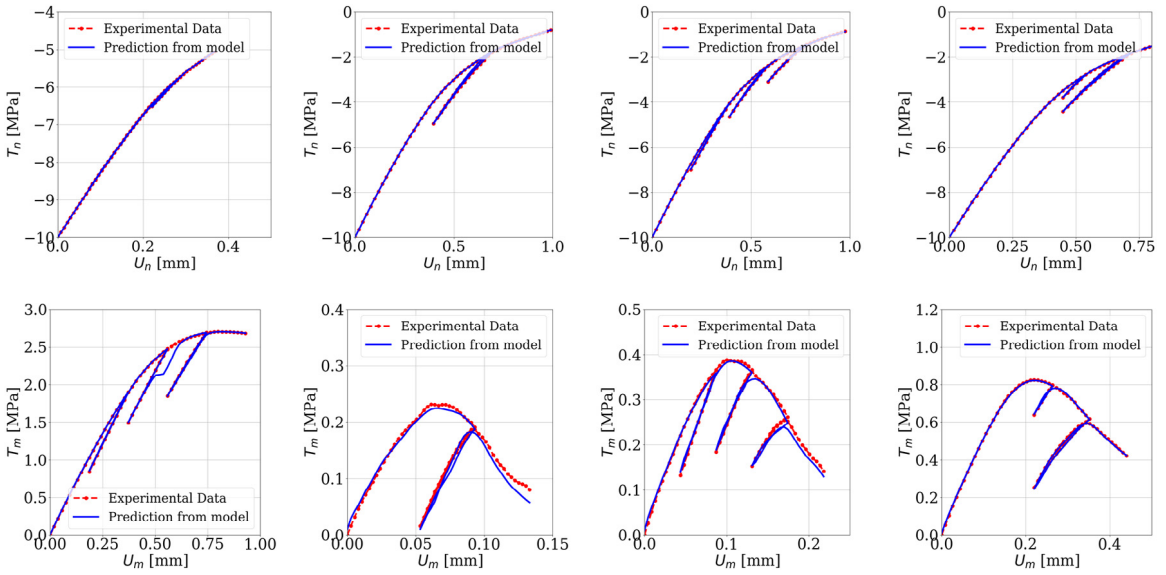
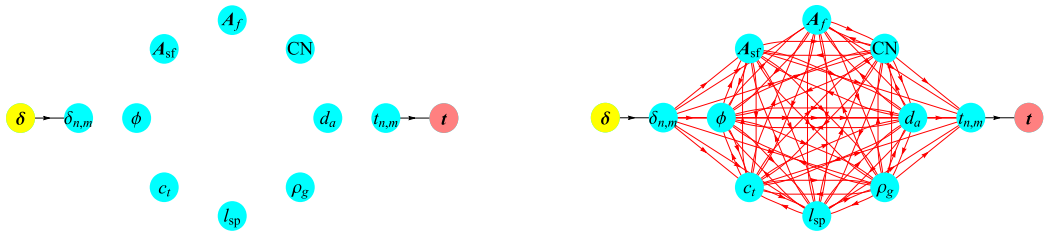


Fig. 9. Four examples of blind predictions from the optimal digraph configuration (the 4th digraph in Fig. 8) against unseen data among test database of 150 loading cases.

experiments, have not yet made significant impacts on constitutive laws used for engineering applications. Our meta-modeling approach is capable of overcoming this bottleneck by efficiently automating some of these tasks currently undertaken by human modelers. This second numerical experiment is used to demonstrate how the augmented intelligence can be used to incorporate new insights from observations into predictions without manually re-writing an existing constitutive law every time new information comes up.

This example is an extension of the first numerical experiment, in which more microstructural information are considered, including the fabric of strong interactions A_{sf} and four measures of grain connectivities $d_a,\ c_t,\ l_{sp},\ \rho_g$. The task of identifying their roles in constitutive models for granular materials is now simply recast as defining a new game with augmented vertex set in digraph and extended action space. The "game board" and all possible actions for this new game are shown in Fig. 10. The dimension of the action space increases from 13 to 71. A particular game rule is added to test the flexibility of the DRL algorithm in handling different types of game constraints: the strong fabric tensor A_{sf} and the fabric tensor A_f, since both are geometric measures of inter-particles forces, are mutually exclusive in the final digraphs of constitutive models. The number of possible game states increases from 3200 to over 400 000. The number of complete and admissible directed graph configurations increases from 591 to over 20 000. The score definition is the same as Eq. (14). The meta-modeling algorithm tries to learn the optimal ways to incorporate the microstructural information and make better predictions only from the training database of 50 loading cases, while the gained knowledge is validated on the test database of another 150 loading cases. The parameters for the DRL meta-modeling algorithm are set as: $numIters = 10$ iterations of "exploration and exploitation", 1 iteration of "competitive gameplay", $numEpisodes = 30$ self-plays in each iteration, and $numMCTSSims = 30$ MCTS simulations in each play step. Hence only a total number of 300 game episodes are needed to complete the DRL procedure, comparing to more than 20 000 possible digraph models in this game setting.

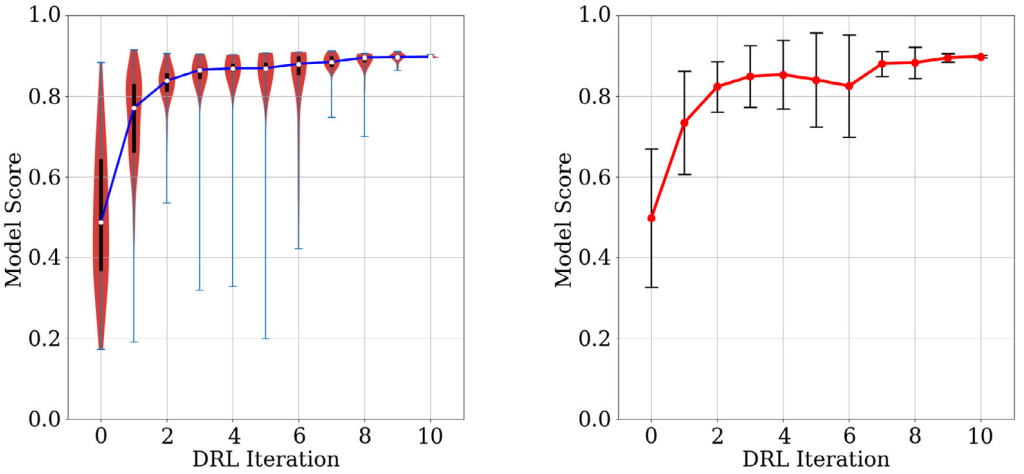
The statistics of the gameplay results from 5 separate runs of the DRL procedure are presented in Fig. 11. We observe a very efficient improvement in generated traction–separation models, even though the number of legal game states in the new game has largely increased. Fig. 12 exhibits four representative digraph configurations developed during DRL iterations, as well as their prediction quality on calibration data and unseen data. It can be seen that the information flows in a constitutive model are of crucial importance. Although the first and the fourth graphs both incorporate the same types of microstructural information, the difference in the ways these information are connected results in significant difference in model scores of 0.191 and 0.915, respectively. Moreover, the DRL algorithm develops the intelligence of selecting the strong fabric tensor A_{sf} over the fabric tensor A_f in order to further improve the prediction score of the model. Four blind prediction examples of the optimal digraph configuration



(a) Initial configuration of the "game board"

(b) All possible actions on the "game board"

Fig. 10. A game of traction–separation model for a digraph involving the nodes $\{\delta_{n,m},\ t_{n,m},\ \phi,\ CN,\ \boldsymbol{A}_f,\ \boldsymbol{A}_{sf},\ d_a,\ c_t,\ l_{sp},\ \rho_g\}$ (detailed in Section 2). (a) The initial "board" on which the game is played. (b) All possible actions for picking the edges connecting the nodes are represented by the red arrows. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)



(a) Violin plots of the density distribution of model scores in each DRL iteration

(b) Mean value and ± standard deviation of model score in each DRL iteration

Fig. 11. Statistics of the model scores in deep reinforcement learning iterations from 5 separate runs of the DRL procedure for Numerical Experiment 2. Each DRL procedure contains ten iterations 0–9 of "exploration and exploitation" (by setting the temperature parameter $\tau = 1.0$) and a final iteration 10 of "competitive gameplay" ($\tau = 0.01$). Each iteration consists of 30 games. (a) Violin Plot of model scores played in each DRL iteration. The shade area represents the density distribution of scores. The white point represents the median. The thick black bar represents the interquartile range between 25% quantile and 75% quantile. The maximum and minimum scores played in each iteration are marked by horizontal lines. (b) Mean model score in each iteration and the error bars mark ± standard deviation.

(the 4th digraph in Fig. 12) obtained in this game are presented in Fig. 13. Comparing to the numerical example 1 (Fig. 9), the augmented knowledge of additional microstructural information in constitutive models lead to more accurate representations of granular materials.

6.3. Application: Multiscale bridging using DRL-generated traction–separation model in finite element modeling

In this example, we demonstrate that the traction–separation model automatically generated by the proposed DRL meta-modeling algorithm can be applied in multiscale finite element simulations, as a surrogate model to replace the computationally expensive DEM RVEs. The example consists of a mixed-mode tension–shear test on a plane-strain granular specimen with embedded strong discontinuity. The sample is 0.1 m × 0.1 m in dimension. The bottom edge is fixed, while the top edge moves rigidly following a mixed-mode loading-unloading-reloading displacement path, as shown in Fig. 14. The sample is assumed periodic in the horizontal direction, thus periodic displacement boundary condition is applied on the lateral edges. The geometry of the pre-existing interface in the center is a sinusoid with the spatial period of 0.05 m and amplitude of 0.005 m. The elements along the interface are enhanced by assumed strain

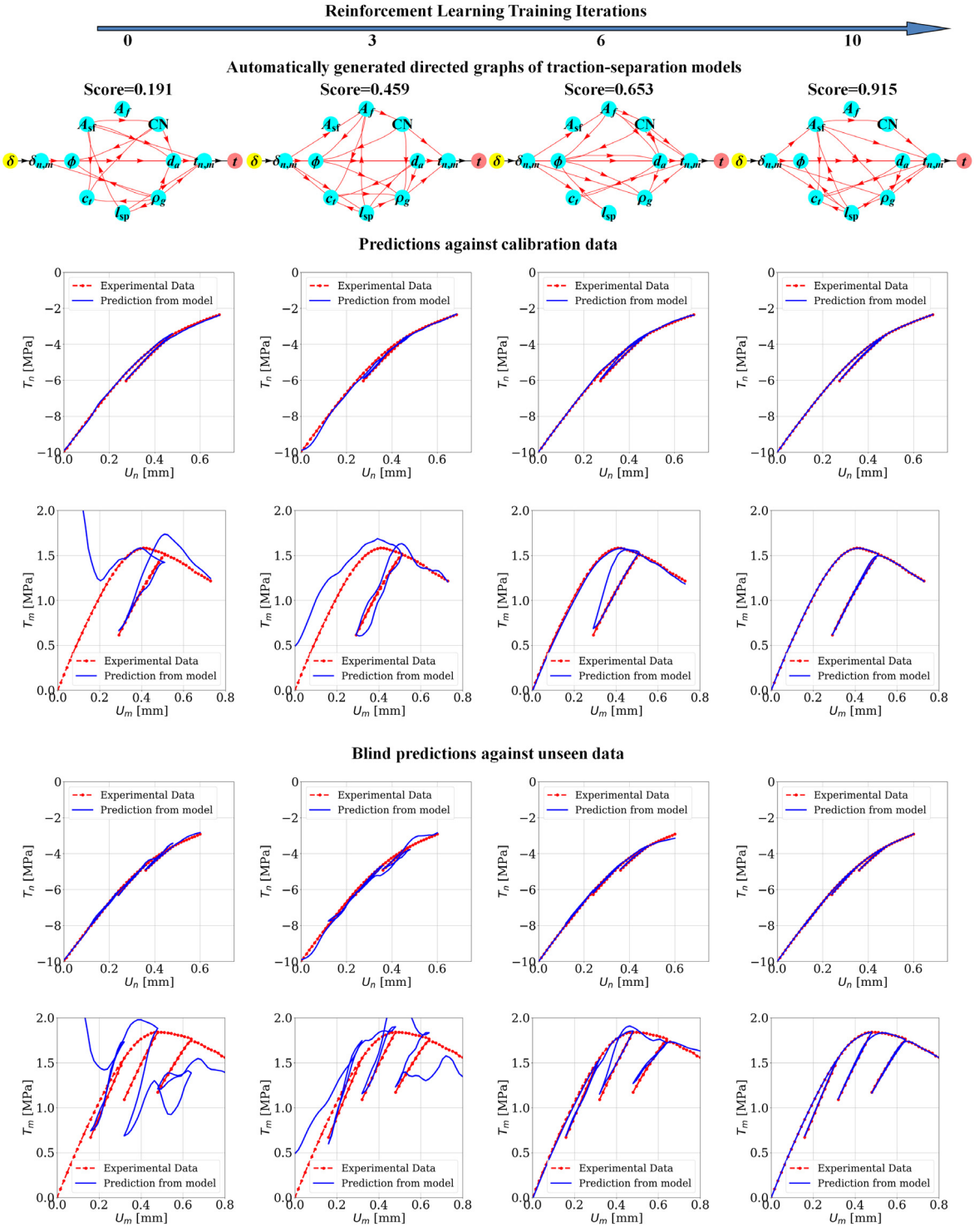


Fig. 12. Knowledge of directed graphs of traction–separation models learned by deep reinforcement learning in Numerical Experiment 2.

formulation to embed the strong discontinuity [82], while the other elements are regular bulk finite elements. The assumed strain formulation requires the solution of the local displacement jump $\boldsymbol{\delta}$ from a local equilibrium equation

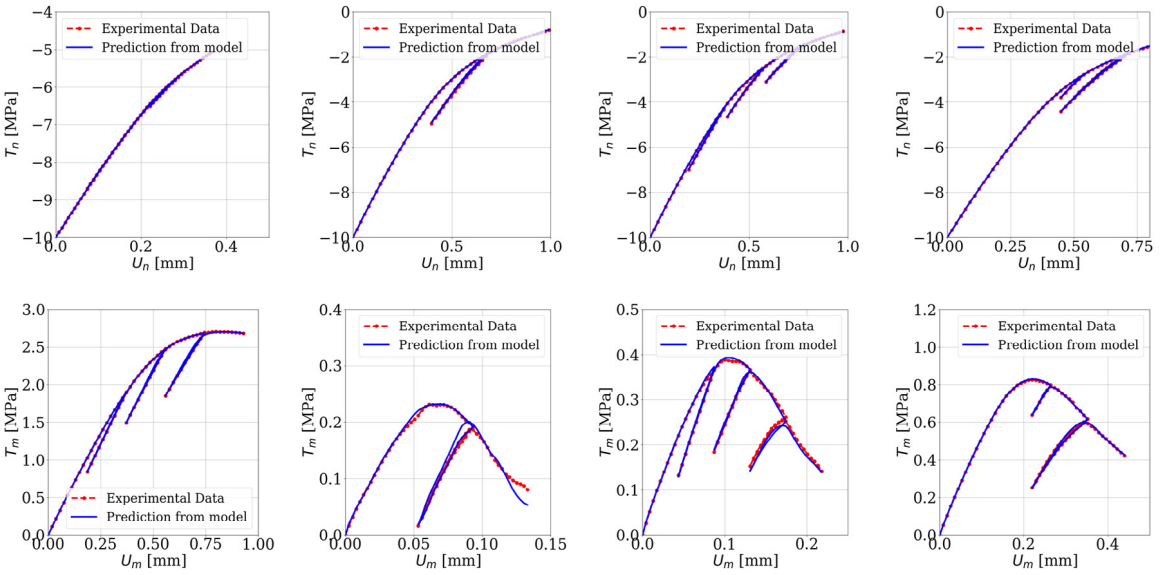
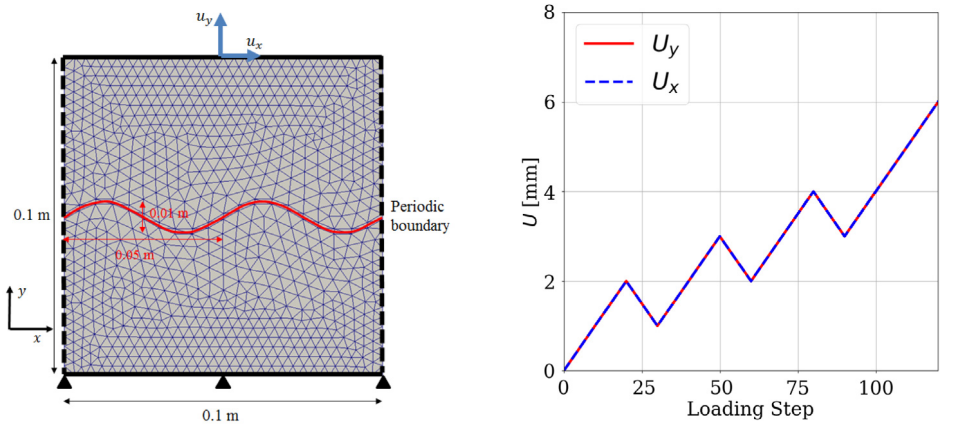


Fig. 13. Four examples of blind predictions from the optimal digraph configuration (The 4th digraph in Fig. 12) against unseen data among test database of 150 loading cases.

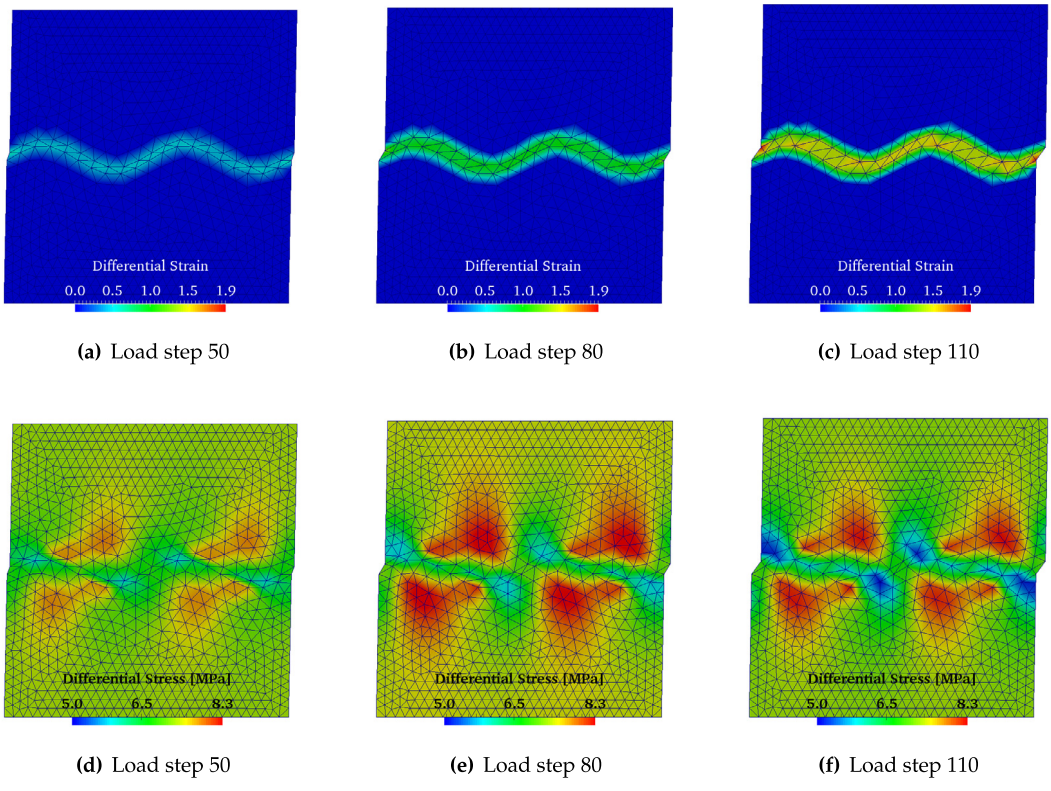


(a) Embedded interface of sinusoidal line **(b)** Prescribed displacement path on the top edge

Fig. 14. Plane-strain mixed-mode tension–shear test on a granular specimen with pre-existing interface. Geometry, mesh and boundary conditions.

linking the traction $\boldsymbol{t}$ across the strong discontinuity and the stress $\boldsymbol{\sigma}$ in the surrounding bulk material in each enhanced element ($\boldsymbol{t}(\boldsymbol{\delta}) = \boldsymbol{\sigma}(\bar{\boldsymbol{\epsilon}}(\boldsymbol{\epsilon}, \boldsymbol{\delta})) \cdot \boldsymbol{n}$, where $\boldsymbol{n}$ is the normal of the interface, $\bar{\boldsymbol{\epsilon}}$ is the continuous strain and $\boldsymbol{\epsilon}$ is the prescribed conformal strain). The optimal traction–separation model found in the Numerical Example 2 (The 4th digraph in Fig. 12) is used to compute the traction $\boldsymbol{t}(\boldsymbol{\delta})$. The computation is executed step-by-step along the information flow in the digraph, starting from the root node $\boldsymbol{\delta}$, and until the leaf node $\boldsymbol{t}$ is reached. Each intermediate node along the information flow is predicted from its predecessor nodes, and predicts its successor nodes. The required ANN for each prediction has already been trained during the model score evaluation procedure in Section 4. The stiffness matrix $\frac{d\boldsymbol{t}}{d\boldsymbol{\delta}}$ is evaluated by the finite difference method. More details on the implementations of ANN models in the assumed strain formulation are presented in [9].

The bulk material is assumed isotropic linear elastic and the parameters are homogenized from the DEM RVE for generation of data (Young's modulus $E = 300$ MPa, Poisson's ratio $\nu = 0.24$). The specimen is initially under



(a) Load step 50 **(b)** Load step 80 **(c)** Load step 110

(d) Load step 50 **(e)** Load step 80 **(f)** Load step 110

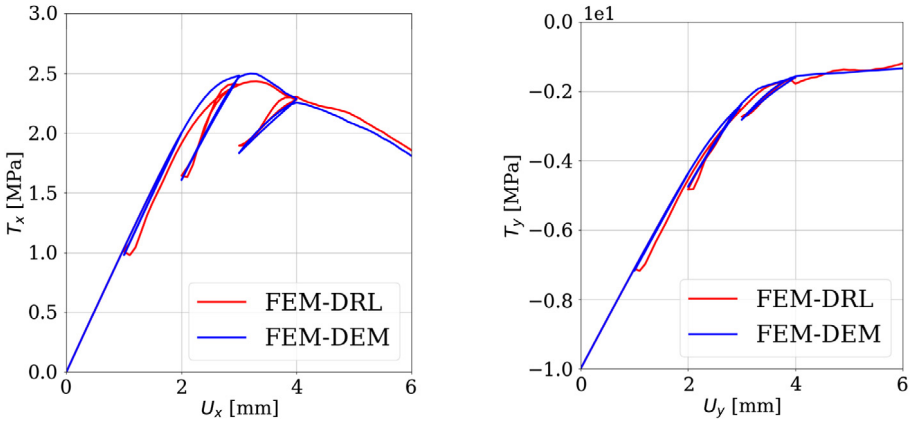
Fig. 15. Differential stress field ($\sigma_1 - \sigma_3$, where σ_1 is the greatest principal value and σ_3 is the least principal value of the Cauchy stress tensor $\boldsymbol{\sigma}$) and differential strain field ($\varepsilon_1 - \varepsilon_3$, where ε_1 is the greatest principal value and ε_3 is the least principal value of the strain tensor $\boldsymbol{\varepsilon}$) at loading steps 50, 80 and 110 in the mixed-mode tension–shear test.

isotropic pressure of 10 MPa, the same as the DEM RVEs, and the lateral pressure remains constant during the loading steps.

The differential stress and strain fields computed by FEM-DRL multiscale approach at selected loading steps are shown in Fig. 15. The shear strain localizes in the embedded strong discontinuity in both specimens. The global traction–displacement curves in normal ($U_y - T_y$) and shear ($U_x - T_x$) directions are compared for both FEM-DRL and FEM-DEM simulations in Fig. 16. Results show great agreements on the mechanical behavior of the specimen. Hence the optimal traction–separation model automatically developed by the DRL meta-modeling approach can be used as a surrogate model to microscale DEM RVEs in multiscale FEM simulations.

7. Conclusion

This paper presents a meta-modeling approach in which we attempt to generate traction–separation laws not through explicitly writing a particular model but to provide the computer with modeling options such that it can explore on its own through self-practicing. Unlike previous deep-learning models that only leverage supervised learning techniques to train neural networks that makes black-box predictions, this new approach focuses on reinforcement learning technique to discover hidden relationships among data and therefore make modeling decisions to emulate the process of writing constitutive models by human. To the best knowledge of the authors, this is the first work on using reinforcement learning and directed graph to form modeling ideas for writing path-dependent constitutive laws. Given the rules (frame indifference, thermodynamic laws, balance principles), we introduce an agenda-based approach where the DRL technique is used to find the optimal way to generate a forward prediction. As demonstrated in our numerical experiments, this approach can be regarded as a generalization of the previous models where neural network predictions may still embed in part of the predictions but are not necessarily completely



(a) Embedded interface of sinusoidal line **(b)** Prescribed displacement path on the top edge

Fig. 16. Comparison of FEM-DRL and FEM-DEM multiscale simulations on global traction–displacement relation in normal ($U_y - T_y$) and tangential ($U_x - T_x$) directions on the top edge of the specimen.

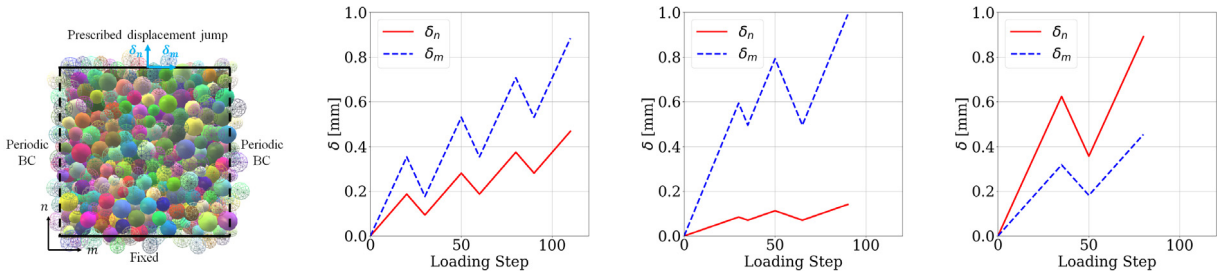
replacing all components in the conventional models. This flexibility is the key for us to exploit the computer to make *repeated* trial-and-errors and improve from experiments over time to generate the best outcomes, instead of spending significant human time to explore through trial-and-errors. The idea of inventing the meta-modeling game could be significant in the sense that it frees modelers from focusing on curve-fitting a physical process. Instead, as future improvements on the models can be made by expanding the action space or simply leveraging the power of computer to improve the models over time, this allows us unprecedented luxury to place our focuses on finding the best cause of actions that lead to the most predictive model. In addition, this meta-modeling approach also provides the following unique benefits against the conventional hand-crafting approach and black-box neural network models.

1. Since the machine learning procedure is automated, models intended for different purposes or designed to fulfill different demands (speed, accuracy, robustness) can be automatically generated and improved over time through self-plays in the model-creation game.
2. Since the validation procedure is introduced as the reward mechanism for the agent to find the optimal models available, the resultant models are always validated at the end of the games.
3. By recasting constitutive models as directed graphs, previous models established by domain experts can be easily embedded in the proposed framework to expand action spaces efficiently and shorten the training time.
4. The meta-modeling approach is generic and reusable, which means that it can handle different situations with different data, objective functions and rules set by human without going through additional derivation, implementation, material parameter identification and validation. Hence it does not require any debugging once the game is implemented correctly.

There are also limitations of the current approaches. For instance, the demands for data could be higher than conventional modeling approaches, particularly when more sophisticated and rigorous validation metric is used to assign model score and game reward. The model has not yet introduced any technique to handle noise, nor does it consider any mechanism to consider uncertainty. While these topics are of great importance, the corresponding research activities are beyond the scope of this study and will be considered in the future, should opportunities come.

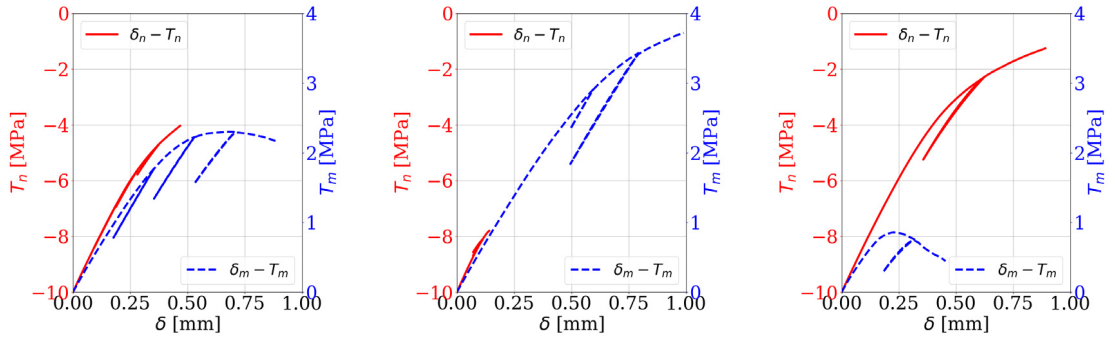
Acknowledgments

The corresponding author's work is supported by the Earth Materials and Processes program from the US Army Research Office under grant contract W911NF-15-1-0442 and W911NF-15-1-0581, the Dynamic Materials and Interactions Program from the Air Force Office of Scientific Research under grant contract FA9550-17-1-0169, the nuclear energy university program from the Department of Energy under grant contract DE-NE0008534 as well as the Mechanics of Material program at National Science Foundation under grant contract CMMI-1462760. Meanwhile,



(a) RVE of frictional surface **(b)** Example loading path 1 **(c)** Example loading path 2 **(d)** Example loading path 3

Fig. 17. Representative volume element of a frictional surface having normal $\boldsymbol{n}$ and tangential $\boldsymbol{m}$ directions. Three examples of different loading-unloading-reloading paths among 200 numerical experiments for the generation of database are shown.



(a) Example loading path 1 **(b)** Example loading path 2 **(c)** Example loading path 3

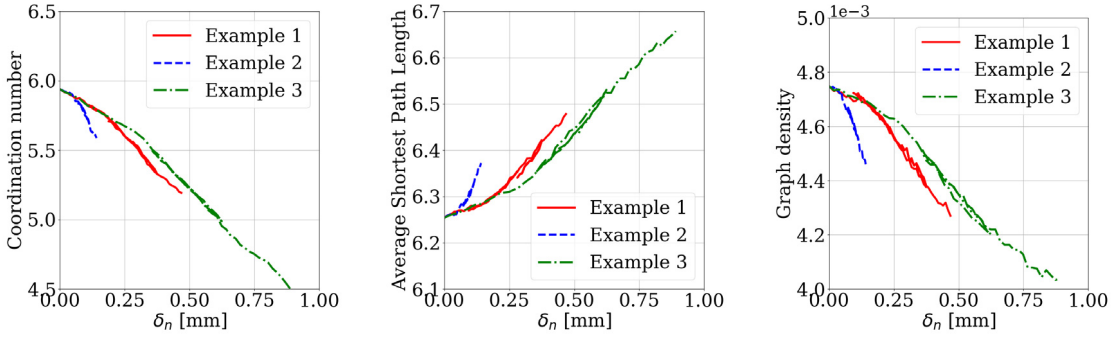
Fig. 18. Examples of traction–separation curves corresponding to the three loading paths in Fig. 17 among 200 numerical simulations. The normal traction T_n is plotted against the normal displacement jump δ_n. The tangential traction T_m is plotted against the tangential displacement jump δ_m.

the first author is supported by US Army Research Office under grant contract W911NF-15-1-0442 and W911NF-15-1-0581, and the 2018 Interdisciplinary Research Seed funding from Columbia University. These supports are gratefully acknowledged. The views and conclusions contained in this document are those of the authors, and should not be interpreted as representing the official policies, either expressed or implied, of the sponsors, including the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

Appendix. Generation of synthetic data from discrete element modeling (DEM)

The data for calibration and evaluation of prediction accuracy of the deep-reinforcement-learned traction–separation models are generated by numerical simulations on a representative volume element (RVE) representing the granular materials on a frictional surface. The open-source software YADE for DEM is used [83]. The discrete element particles in the RVE have radii between 1 ± 0.3 mm with uniform distribution. The RVE has the height of 20 mm in the normal direction of the frictional surface and is initially consolidated to isotropic pressure of 10 MPa. The Cundall's elastic-frictional contact model [84] is used for the inter-particle constitutive law. The material parameters are: interparticle elastic modulus $E_{eq} = 1$ GPa, ratio between shear and normal stiffness $k_s/k_n = 0.3$, frictional angle $\varphi = 30°$, density $\rho = 2600$ kg/m^3, Cundall damping coefficient $\alpha_{damp} = 0.2$.

The DEM RVE is loaded in the normal $\boldsymbol{n}$ and tangential $\boldsymbol{m}$ directions of the frictional surface by displacement controls δ_n and δ_m (Fig. 17(a)). The synthetic database consists of 200 numerical experiments under different loading paths. They differ from each other in the ratio of normal and tangential loading rate $\dot{\delta_n}/\dot{\delta_m}$, as well as the loading-unloading-reloading cycles, as illustrated in Figs. 17(b)–17(d). The traction–separation curves of the experiments are recorded and three examples corresponding to the paths in Fig. 17 are presented in Fig. 18. The microstructural information required for the intermediate nodes in the directed graphs, such as porosity, coordination number and



(a) Coordination number **(b)** Average Shortest Path Length **(c)** Graph density

Fig. 19. Examples of coordination number, average shortest path length and graph density for particle interactions corresponding to the three loading paths in Fig. 17 among 200 numerical simulations. These quantities are plotted against the normal displacement jump δ_n.

fabric tensor, are also recorded during the simulations. The open-source library NetworkX [56] is employed to analyze the graph of the particle interactions in the RVEs. Fig. 19 presents examples of microstructural information and graph characteristics for the three example loading paths. The 200 numerical simulations in the database are shuffled. The first 50 simulations are used as "training data" for the calibration of model parameters for the edges in the directed graphs. The other 150 simulations are "test data" only for evaluating the blind prediction accuracy of the resultant constitutive model.

References

[1] James R. Rice, A path independent integral and the approximate analysis of strain concentration by notches and cracks, J. Appl. Mech. 35 (2) (1968) 379–386.
[2] Kyoungsoo Park, Glaucio H. Paulino, Jeffery R. Roesler, A unified potential-based cohesive model of mixed-mode fracture, J. Mech. Phys. Solids 57 (6) (2009) 891–908.
[3] Kun Wang, WaiChing Sun, A unified variational eigen-erosion framework for interacting brittle fractures and compaction bands in fluid-infiltrating porous media, Comput. Methods Appl. Mech. Engrg. 318 (2017) 1–32.
[4] Eric C. Bryant, WaiChing Sun, A mixed-mode phase field fracture model in anisotropic rocks with consistent kinematics, Comput. Methods Appl. Mech. Engrg. (ISSN: 0045-7825) (2018) http://dx.doi.org/10.1016/j.cma.2018.08.008, URL http://www.sciencedirect.com/science/article/pii/S0045782518303943.
[5] Timon Rabczuk, P.M.A. Areias, A new approach for modelling slip lines in geological materials with cohesive models, Int. J. Numer. Anal. Methods Geomech. 30 (11) (2006) 1159–1172.
[6] Ronaldo I. Borja, Craig D. Foster, Continuum mathematical modeling of slip weakening in geological systems, J. Geophys. Res. Solid Earth 112 (B4) (2007).
[7] M.G.G.V. Elices, G.V. Guinea, J. Gomez, J. Planas, The cohesive zone model: advantages, limitations and challenges, Eng. Fract. Mech. 69 (2) (2002) 137–163.
[8] Mitiyasu Ohnaka, Teruo Yamashita, A cohesive zone model for dynamic shear faulting based on experimentally inferred constitutive relation and strong motion source parameters, J. Geophys. Res. Solid Earth 94 (B4) (1989) 4089–4104.
[9] Kun Wang, WaiChing Sun, A multiscale multi-permeability poroplasticity model linked by recursive homogenizations and deep learning, Comput. Methods Appl. Mech. Engrg. 334 (2018) 337–380.
[10] WaiChing Sun, Teng-fong Wong, Prediction of permeability and formation factor of sandstone with hybrid lattice Boltzmann/finite element simulation on microtomographic images, Int. J. Rock Mech. Mining Sci. 106 (2018) 269–277.
[11] Michael Ortiz, Anna Pandolfi, Finite-deformation irreversible cohesive elements for three-dimensional crack-propagation analysis, Internat. J. Numer. Methods Engrg. 44 (9) (1999) 1267–1282.
[12] John W. Rudnicki, Fracture mechanics applied to the Earth's crust, Ann. Rev. Earth Planet. Sci. 8 (1) (1980) 489–525.
[13] Mervyn S. Paterson, Teng-fong Wong, Experimental Rock Deformation-The Brittle Field, Springer Science & Business Media, 2005.
[14] WaiChing Sun, A unified method to predict diffuse and localized instabilities in sands, Geomech. Geoeng. 8 (2) (2013) 65–75.
[15] Ronaldo I. Borja, Plasticity: modeling & computation, Springer Science & Business Media, 2013.
[16] Kun Wang, Waiching Sun, Simon Salager, S. Na, Ghonwa Khaddour, Identifying material parameters for a micro-polar plasticity model via x-ray micro-ct images: lessons learned from the curve-fitting exercises, Int. J. Multiscale Comput. Eng. 14 (4) (2016) 389–413.
[17] SeonHong Na, WaiChing Sun, Computational thermo-hydro-mechanics for multiphase freezing and thawing porous media in the finite deformation range, Comput. Methods Appl. Mech. Engrg. 318 (2017) 667–700.

[18] Nicolas Moës, Mathieu Cloirec, Patrice Cartraud, J-F Remacle, A computational approach to handle complex microstructure geometries, Comput. Methods Appl. Mech. Engrg. 192 (28–30) (2003) 3163–3177.
[19] Claudia Britta Hirschberger, Natarajan Sukumar, Paul Steinmann, Computational homogenization of material layers with micromorphic mesostructure, Phil. Mag. 88 (30–32) (2008) 3603–3631.
[20] C.B. Hirschberger, S. Ricker, P. Steinmann, N. Sukumar, Computational multiscale modelling of heterogeneous material layers, Eng. Fract. Mech. 76 (6) (2009) 793–812.
[21] Frédéric Feyel, A multilevel finite element method (FE2) to describe the response of highly non-linear structures using generalized continua, Comput. Methods Appl. Mech. Engrg. 192 (28–30) (2003) 3233–3244.
[22] WaiChing Sun, José E. Andrade, John W. Rudnicki, Peter Eichhubl, Connecting microstructural attributes and permeability from 3D tomographic images of in situ shear-enhanced compaction bands using multiscale computations, Geophys. Res. Lett. 38 (10) (2011).
[23] Wai Ching Sun, Jose E. Andrade, John W. Rudnicki, Multiscale method for characterization of porous microstructures and their impact on macroscopic effective permeability, Internat. J. Numer. Methods Engrg. 88 (12) (2011) 1260–1279.
[24] Jacob Fish, Practical Multiscaling, John Wiley & Sons, 2013.
[25] WaiChing Sun, Matthew R. Kuhn, John W. Rudnicki, A multiscale DEM-LBM analysis on permeability evolutions inside a dilatant shear band, Acta Geotech. 8 (5) (2013) 465–480.
[26] Kun Wang, WaiChing Sun, Anisotropy of a tensorial Bishop's coefficient for wetted granular materials, J. Eng. Mech. 143 (3) (2015) B4015004.
[27] Yang Liu, WaiChing Sun, Zifeng Yuan, Jacob Fish, A nonlocal multiscale discrete-continuum model for predicting mechanical behavior of granular materials, Internat. J. Numer. Methods Engrg. 106 (2) (2016) 129–160.
[28] Matthew R. Kuhn, WaiChing Sun, Qi Wang, Stress-induced anisotropy in granular materials: fabric, stiffness, and permeability, Acta Geotech. 10 (4) (2015) 399–419.
[29] Kun Wang, WaiChing Sun, A semi-implicit discrete-continuum coupling method for porous media based on the effective stress principle at finite strain, Comput. Methods Appl. Mech. Engrg. 304 (2016) 546–583.
[30] Huanran Wu, Ning Guo, Jidong Zhao, Multiscale modeling and analysis of compaction bands in high-porosity sandstones, Acta Geotech. 13 (3) (2018) 575–599.
[31] Kedar Kirane, Somnath Ghosh, A cold dwell fatigue crack nucleation criterion for polycrystalline Ti-6242 using grain-level crystal plasticity FE model, Int. J. Fatigue 30 (12) (2008) 2127–2139.
[32] Clemens V. Verhoosel, Joris J.C. Remmers, Miguel A. Gutiérrez, Rene De Borst, Computational homogenization for adhesive and cohesive failure in quasi-brittle solids, Internat. J. Numer. Methods Engrg. 83 (8–9) (2010) 1155–1179.
[33] Shahriyar Keshavarz, Somnath Ghosh, Multi-scale crystal plasticity finite element model approach to modeling nickel-based superalloys, Acta Mater. 61 (17) (2013) 6549–6561.
[34] Jitesh H. Panchal, Surya R. Kalidindi, David L. McDowell, Key computational modeling issues in integrated computational materials engineering, Comput. Aided Des. 45 (1) (2013) 4–25.
[35] Tanvir R. Faisal, Nicolay Hristozov, Tamara L. Western, Alejandro D. Rey, Damiano Pasini, Computational study of the elastic properties of Rheum rhabarbarum tissues via surrogate models of tissue geometry, J. Struct. Biol. 185 (3) (2014) 285–294.
[36] Yang Liu, WaiChing Sun, Jacob Fish, Determining material parameters for critical state plasticity models based on multilevel extended digital database, J. Appl. Mech. 83 (1) (2016) 011003.
[37] Aaron E. Tallman, Laura P. Swiler, Yan Wang, David L. McDowell, Reconciled top-down and bottom-up hierarchical multiscale calibration of bcc fe crystal plasticity, Int. J. Multiscale Comput. Eng. 15 (6) (2017).
[38] B.A. Le, Julien Yvonnet, Q.-C. He, Computational homogenization of nonlinear elastic materials using neural networks, Internat. J. Numer. Methods Engrg. 104 (12) (2015) 1061–1084.
[39] M.A. Bessa, R. Bostanabad, Z. Liu, A. Hu, Daniel W. Apley, C. Brinson, W. Chen, Wing Kam Liu, A framework for data-driven analysis of materials under uncertainty: countering the curse of dimensionality, Comput. Methods Appl. Mech. Engrg. 320 (2017) 633–667.
[40] Daniele Versino, Alberto Tonda, Curt A. Bronkhorst, Data driven modeling of plastic deformation, Comput. Methods Appl. Mech. Engrg. 318 (2017) 981–1004.
[41] Orion L. Kafka, Cheng Yu, Modesar Shakoor, Zeliang Liu, Gregory J. Wagner, Wing Kam Liu, Data-driven mechanistic modeling of influence of microstructure on high-cycle fatigue life of nickel titanium, JOM (2018) 1–5.
[42] Stephan Wulfinghoff, Fabiola Cavaliere, Stefanie Reese, Model order reduction of nonlinear homogenization problems using a Hashin–Shtrikman type finite element method, Comput. Methods Appl. Mech. Engrg. 330 (2018) 149–179.
[43] M. Lefik, B.A. Schrefler, Artificial neural network for parameter identifications for an elasto-plastic model of superconducting cable under cyclic loading, Comput. Struct. 80 (22) (2002) 1699–1713.
[44] Sinno Jialin Pan, Qiang Yang, et al., A survey on transfer learning, IEEE Trans. Knowl. Data Eng. 22 (10) (2010) 1345–1359.
[45] Richard S. Sutton, Introduction: The challenge of reinforcement learning, in: Reinforcement Learning, Springer, 1992, pp. 1–3.
[46] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al., Mastering the game of Go with deep neural networks and tree search, nature 529 (7587) (2016) 484–489.
[47] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al., Mastering the game of Go without human knowledge, Nature 550 (7676) (2017) 354.
[48] Claude E. Shannon, XXII. programming a computer for playing chess, London Edinburgh Dublin Phil. Mag. J. Sci. 41 (314) (1950) 256–275.
[49] Richard Bellman, A Markovian decision process, J. Math. Mech. (1957) 679–684.
[50] ICapuzzo Dolcetta, Hitoshi Ishii, Approximate solutions of the Bellman equation of deterministic control theory, Appl. Math. Optim. 11 (1) (1984) 161–181.

[51] WaiChing Sun, Jakob T. Ostien, Andrew G. Salinger, A stabilized assumed deformation gradient finite element formulation for strongly coupled poromechanical simulations at finite strain, Int. J. Numer. Anal. Methods Geomech. 37 (16) (2013) 2755–2788.
[52] WaiChing Sun, Qiushi Chen, Jakob T. Ostien, Modeling the hydro-mechanical responses of strip and circular punch loadings on water-saturated collapsible geomaterials, Acta Geotech. 9 (5) (2014) 903–934.
[53] WaiChing Sun, A stabilized finite element formulation for monolithic thermo-hydro-mechanical simulations at finite strain, Internat. J. Numer. Methods Engrg. 103 (11) (2015) 798–839.
[54] Andrew G. Salinger, Roscoe A. Bartlett, Andrew M. Bradley, Qiushi Chen, Irina P. Demeshko, Xujiao Gao, Glen A. Hansen, Alejandro Mota, Richard P. Muller, Erik Nielsen, et al., Albany: using component-based design to develop a flexible, generic multiphysics analysis code, Int. J. Multiscale Comput. Eng. 14 (4) (2016).
[55] Jørgen Bang-Jensen, Gregory Z. Gutin, Digraphs: Theory, Algorithms and Applications, Springer Science & Business Media, 2008.
[56] Aric Hagberg, Pieter Swart, Daniel S.C.hult, Exploring network structure, dynamics, and function using NetworkX, Technical report, Los Alamos National Lab. (LANL), Los Alamos, NM (United States), 2008.
[57] Maurice George Kendall, et al., The advanced theory of statistics, in: The advanced theory of statistics, second ed., Charles Griffin and Co., Ltd., London, 1946.
[58] Theodore W. Anderson, Donald A. Darling, A test of goodness of fit, J. Amer. Statist. Assoc. 49 (268) (1954) 765–769.
[59] Fritz W. Scholz, Michael A. Stephens, K-sample anderson–darling tests, J. Amer. Statist. Assoc. 82 (399) (1987) 918–924.
[60] Kurt Hornik, Maxwell Stinchcombe, Halbert White, Multilayer feedforward networks are universal approximators, Neural Netw. 2 (5) (1989) 359–366.
[61] Sepp Hochreiter, Jürgen Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.
[62] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, Yoshua Bengio, Learning phrase representations using RNN encoder-decoder for statistical machine translation, 2014, arXiv preprint arXiv:1406.1078.
[63] François Chollet, et al., Keras, 2015, https://keras.io.
[64] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: machine learning in python, J. Mach. Learn. Res. 12 (2011) 2825–2830.
[65] Diederik P. Kingma, Jimmy Ba, Adam: a method for stochastic optimization, 2014, arXiv preprint arXiv:1412.6980.
[66] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al., Mastering chess and shogi by self-play with a general reinforcement learning algorithm, 2017, arXiv preprint arXiv:1712.01815.
[67] Cameron B. Browne, Edward Powley, Daniel Whitehouse, Simon M. Lucas, Peter I. Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, Simon Colton, A survey of monte carlo tree search methods, IEEE Tran. Comput. Intell. AI Games 4 (1) (2012) 1–43.
[68] Nasser M. Nasrabadi, Pattern recognition and machine learning, J. Electron. Imaging 16 (4) (2007) 049901.
[69] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al., Human-level control through deep reinforcement learning, Nature 518 (7540) (2015) 529.
[70] Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al., Relational inductive biases, deep learning, and graph networks, 2018, arXiv preprint arXiv:1806.01261.
[71] Pengcheng Fu, Yannis F. Dafalias, Fabric evolution within shear bands of granular materials and its relation to critical state theory, Int. J. Numer. Anal. Methods Geomech. 35 (18) (2011) 1918–1948.
[72] Xiang Song Li, Yannis F. Dafalias, Anisotropic critical state theory: role of fabric, J. Eng. Mech. 138 (3) (2011) 263–275.
[73] M. Pastor, A.H.C. Chan, P. Mira, D. Manzanal, J.A. Fernández Merodo, T. Blanc, Computational geomechanics: the heritage of Olek Zienkiewicz, Internat. J. Numer. Methods Engrg. 87 (1–5) (2011) 457–489.
[74] Stephen Timoshenko, History of strength of materials: with a brief account of the history of theory of elasticity and theory of structures, Courier Corporation, 1953.
[75] Morteza M. Mehrabadi, S. Nemat-Nasser, M. Oda, On statistical description of stress and fabric in granular materials, Int. J. Numer. Anal. Methods Geomech. 6 (1) (1982) 95–108.
[76] Yannis F. Dafalias, Majid T. Manzari, Simple plasticity sand model accounting for fabric change effects, J. Eng. Mech. 130 (6) (2004) 622–634.
[77] Yannis F. Dafalias, Achilleas G. Papadimitriou, Xiang S. Li, Sand plasticity model accounting for inherent fabric anisotropy, J. Eng. Mech. 130 (11) (2004) 1319–1333.
[78] Antoinette Tordesillas, David M. Walker, Amy L. Rechenmacher, Sara Abedi, Discovering community structures and dynamical networks from grain-scale kinematics of shear bands in sand, in: Advances in Bifurcation and Degradation in Geomaterials, Springer, 2011, pp. 67–73.
[79] Antoinette Tordesillas, David M. Walker, Qun Lin, Force cycles and force chains, Phys. Rev. E 81 (1) (2010) 011302.
[80] John R. Williams, Nabha Rege, Coherent vortex structures in deforming granular materials, Mech. Cohesive-frictional Mater. Int. J. Exp. Model. Comput. Mater. Struct. 2 (3) (1997) 223–236.
[81] Guang Liu, WaiChing Sun, Steven M Lowinger, ZhenHua Zhang, Ming Huang, Jun Peng, Coupled flow network and discrete element modeling of injection-induced crack propagation and coalescence in brittle rock, Acta Geotech. (2018) 1–26.
[82] J. Oliver, Alfredo Edmundo Huespe, M.D.G. Pulido, E. Chaves, From continuum mechanics to fracture mechanics: the strong discontinuity approach, Eng. Fract. Mech. 69 (2) (2002) 113–136.
[83] Václav Šmilauer, Emanuele Catalano, Bruno Chareyre, Sergei Dorofeenko, Jerome Duriez, Anton Gladky, Janek Kozicki, Chiara Modenese, Luc Scholtès, Luc Sibille, et al., Yade reference documentation, Yade Documentation 474 (1) (2010).
[84] Peter A. Cundall, Otto D.L. Strack, A discrete numerical model for granular assemblies, Geotechnique 29 (1) (1979) 47–65.